2006

Extraction and interaction analysis of foreground objects in panning video

Raja Jain

Extraction and Interaction Analysis of Foreground Objects in Panning Video

by

Raja P. Jain

Thesis report submitted in partial fulfillment of the requirements of the degree of
Master of Science in Computer Science

Department of Computer Science
B.Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York
May 2006

Approved By:

Dr. Roger S. Gaborski
Professor, Department of Computer Science
Advisor

Dr. Carl H. Reynolds
Professor, Department of Computer Science
Reader

Dr. Peter G. Anderson
Professor Emeritus, Department of Computer Science
Observer

Acknowledgements

I am thankful to my parents and brother for supporting me through my master's education. The faculties of the Computer Science department of the Rochester Institute of Technology carefully guided me through my Master's Degree program. I am grateful to Dr. Roger Gaborski for providing me with guidance and support during the thesis process. He helped me considerably in building my concepts in the field of Computer Vision, gave me good ideas, and encouraged me at every step. I am also thankful to Dr. Carl Reynolds for being my reader and guiding me during thesis procedure. I would also like to thank Dr. Peter Anderson for being my observer.

Abstract

In this work, moving foreground objects are extracted from video. The video camera can be either stationary or panning. The motion information extracted from consecutive frames of video is used for object segmentation, and an appearance model of the moving object is created. No assumptions are made concerning the object model, resulting in segmented objects of varying shapes and sizes. Finally, relationships between objects in the image are evaluated toward the goal of higher level understanding of videos.

Objectives

1) Extracting moving foreground objects in a video taken from either stationary or moving camera.
2) Finding the relative velocity of the moving foreground objects in the video.
3) Segmenting the moving foreground objects using motion, distance, and object relationships obtained from few consecutive frames of video. The color information doesn't play a major role in object segmentation, resulting in a system which is able to segment foreground objects having very similar color and texture as that of the background.
4) Generating appearance model of the moving objects, i.e., determining the ranges of colors present in moving objects. The algorithm is able to determine what colors are present in the segmented moving foreground objects. The system can search and highlight the objects having specified color and can also highlight the particular part of the objects having the specified color.
5) Extracting the spatial relationship between any two segmented moving objects. This is a step towards higher level understanding of the videos. Based on this work, a system can be made which can search the videos based on the queries specifying the spatial relationships among the objects such as, "find the videos where 2 objects meet each other", "find the videos in which one object is following the other object", "find the videos where 2 objects meet each other and the go away/together", etc.
6) There is no assumption of the object model, resulting in the segmentation of the objects of varying shapes and sizes.

Table of Contents

1. Introduction

1.1 Moving Object Segmentation

Segmenting moving objects in video is a fundamental step in the process of developing an automated computer vision system for higher level understanding of the videos. For achieving higher level understanding of videos, the system should be able to extract interesting objects from the sequence of frames and should be able to develop a content-based description of video, establish relationships between the interesting objects and generate meaningful information from them. It can have a variety of applications such as video surveillance, gesture recognition, traffic monitoring, etc. Examples of video surveillance are detecting suspicious behavior in parking lots, interaction between people in public places such as restaurants, and speeding and suspiciously moving cars, etc. Once object segmentation is accomplished, a system for object recognition can be made on the top of it and the system can be used in applications such as regularizing pedestrians and highway traffic using density evaluations obtained from segmenting moving people and vehicles.

Also, the recent techniques in image/video coding, segment image/videos into objects to and achieve high compression by separately coding the contour and texture of those objects. Since the sole purpose is to achieve efficient compression, the segmented objects may not be semantically meaningful to humans. Other applications such as content based video retrieval require the segmented objects to be semantically meaningful to the human observers.

An ideal object segmentation system should be able to identify the semantically meaningful components of the image and should be able to group the pixels belonging to such components. It is difficult to segment the static objects in the video, but moving objects can be segmented by using their motion information along with other information.

1.2 Definition of the Problem

This work addresses the problem of extracting moving objects from the video taken with either stationary or panning camera, generating their appearance model, and trying to establish spatial relationships among them. No assumptions are made concerning the object model, so the system should be able to detect moving objects of varying shapes and sizes. Moving objects are segmented primarily from the motion-based information and the relationship they maintain over a certain number of frames. Since the segmentation is not based on color, it is able to distinguish between moving foreground objects and background of almost the same color and texture. Also the system is able to generate appearance models of the moving objects. For example, if the scene contains a huge moving elephant and a very small moving person, since there is no assumption involved regarding the size of the moving objects, the system is successfully able to segment the moving elephant and the small moving person. Also, if the video contains multicolor objects, e.g., a moving car of various colors, then the system is able to

segment it since color information doesn't play a major role in object segmentation. Also the system is able to tell what ranges of color are present in the car, i.e., it generates the appearance model of the car.

The information such as location of objects in each frame, ranges of color present in the object, relative velocity of the objects, and the number of frame when the objects entered and left the scene can be used to generate spatial relationships between the objects. Generating the spatial relationship among the moving foreground objects is a step towards higher level understanding of the videos.

The source code was developed using Intel's OpenCV library [30] in Microsoft's VC++ 6.0 IDE on Windows XP platform.

1.3 Thesis Outline

In chapter 1, definition of the problem and application of the problem in various fields of computer vision is explained.

The first step in solving the problem is background subtraction. It is explained in detail in chapter 2. Section 2.1 explains the need and the definition of the problem of background subtraction. Section 2.2 elaborates on the related work done in the field of background subtraction, their drawbacks and the reason why they can't be used in this thesis problem. Section 2.3 and its subsections establish the background required to explain the approach taken in this thesis for background subtraction. Section 2.4 gives the block diagram explaining the implementation of the algorithm for background subtraction. Section 2.5 shows the results obtained after background subtraction.

Chapter 3 explains the definition of the problem of object segmentation, its purpose, existing algorithms to solve the problem, and overview of the novel approach used in this thesis to solve the problem of object segmentation. Section 3.1 explains in detail the problem of object segmentation and its need in the applications of computer vision. Section 3.2 and its subsections classify different algorithms available for segmenting moving objects, give a detailed explanation of the existing algorithms, explains the drawbacks and advantages of different approaches, and provides the references to different materials required for further in depth study of these approaches. Section 3.3 gives a block diagram providing the overview of the algorithm used in this work for solving the problem of object segmentation.

Chapters 4, 5 and 6 explain in detail the different levels of the block diagram shown in the section 3.3.

Chapter 4 introduces the concept of ***'motion list',*** which is used to effectively store the pixels on the silhouettes of the moving objects and is very effective in reducing the time complexity required in computing the relative velocity of the moving objects. Section 4.1 establishes the background required to appreciate the idea of constructing the ***'motion list'***. Section 4.2 explains in detail the structure of ***'motion list'*** and explains the purpose

and use of its member fields. Section 4.3 explains in detail the existing methods for finding relative velocity of the moving objects in the video. Section 4.4 explains the approach used in this work for finding the relative velocity of the moving pixels. Section 4.5 explains in detail the advantages of using ***'motion list'***. Section 4.6 shows the results obtained after generating the motion list.

Chapter 5 introduces the concept of ***'region list'***. The nodes in ***'motion list'*** are clustered to obtain a ***'region list'.*** Section 5.1 describes in detail the structure of ***'region list'*** and its members. It also explains the purpose of its members. Section 5.2 explains in detail the parameters considered for clustering the nodes of ***'motion list'*** into ***'region list'***. Section 5.3 shows the resulting regions on the few video frames.

Chapter 6 introduces the concept of ***'object list'.*** Section 6.1 explains the construction of the ***'object list'***. Regions formed by clustering the nodes in ***'motion list'*** are stored in a '***region list',*** and are further clustered into the objects, which in turn might represent a part of a complete object. Section 6.2 explains the structure of ***'object list'***, its members and their function. Section 6.3 explains the parameters considered for clustering the nodes in the ***'region list'*** into an ***'object list'***. Section 6.4 explains the role of intensity during different levels of clustering. Section 6.5 explains how the relationships among the different objects in the ***'object list'***, for a certain number of consecutive frames, are tracked and how the related objects are grouped into semantically single large object. It also explains the algorithm in the form of a block diagram. Section 6.6 shows the results of the final stage of object segmentation. Section 6.7 describes a post processing method of merging bounding boxes, to improve the segmentation results on moving non-rigid bodies.

Chapter 7 describes in detail the procedure for generating appearance model for the segmented moving objects.

Chapter 8 concludes the work done in this thesis and explains some of the drawbacks and how they can be addressed in future.

Chapter 9 gives references to different works used during the thesis procedure.

2. Background Subtraction

2.1 Introduction

Extracting moving foreground objects from video sequences is the basic step in many video processing applications involving surveillance, because background is generally considered uninteresting due to its static nature. Once the moving foreground objects are detected, the further processing for tracking and activity is limited in the corresponding regions of the image. In vision-based systems, such detection is usually carried out by using background subtraction methods. Different algorithms can be employed depending on if the camera is stationary or moving.

The simplest case is when the camera is stationary. For the video taken with a fixed camera most background subtraction methods build a model of the scene background, and for each pixel in the image, detect deviations of pixel feature values from the model to classify the pixel as belonging either to background or foreground. Few works like [3] have also combined the pixel intensity information with edge information to obtain better scene modeling.

But the difficult case is when the camera is panning. For video sequences taken from a moving camera both background and foreground objects are in relative motion. So we cannot have any background model, and simple background differencing methods wouldn't work.

2.2 Related Work

For a static camera, the simplest method is to compare the current observed state with an empty background scene. But this solution does not adapt to the dynamic changes in background or changes in lightning conditions. Stauffer and Grimson [1] proposed a background model in which every pixel is represented by a mixture of Gaussians and the model is updated by using an online approximation. This method takes into account the bi-modal or multi-modal background or persistent scene changes. Power and Schoones [2] provide the values for all the parameters of the background model described in [1] and also describes in detail how the background model in [1] is practically implemented.

The algorithm proposed in [1] and [2] is based on the idea that a non-random observation will fall within a standard deviation of an average. Power and Schoones [2] use three Gaussian models for each pixel, where each Gaussian model has three Gaussian distributions, one for each RGB color plane.

In statistics, **Gaussian distribution** is also called ***'normal distribution'***. It is a bell-shaped curve represented by two variables, mean (μ), and standard deviation (σ). Using μ and σ, the Gaussian distributions can be plotted and applied to various probability functions to derive the likelihood that a given observation is an element of that distribution. In [1] they use one Gaussian Model for each pixel of video which in turn contains 3 Gaussian distributions, one for each component of the RGB color space. Each

Gaussian model has a weight (ω) which is used to account for the relative amount of time that a given Gaussian model has represented a particular pixel. It is updated for each Gaussian model at every step. The weight for a frequently observed Gaussian model would be higher than the weight of the less frequently observed Gaussian model.

Probability density defines the likelihood that a given pixel belongs to a given Gaussian model. The equation they use is [1]

$$\eta(X,\mu,\Sigma)=\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}e^{\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)}$$

A **multi-modal** background has more that one background state. For example, flashing highlights on water, bright and dark sides of leaves in the wind, traffic lights have color when turned on and don't have color when turned off. A **bi-modal** background is a specific case of multi-modal background where the background object has only two states.

The algorithm in [1] and [2] runs in a pixel by pixel manner and the authors assume **temporal locality** of pixels, which states that a pixel will most likely be relatively static over close increments of time. They assume that the foreground pixels will vary from the background pixels. Since the background pixels are static, when a background pixel is observed, it will be very close to the other observation. Most observations of the foreground objects, for example a driving car or a moving elephant on brownish ground will vary from the background. For a multi-modal background, a pixel is assigned to the closest Gaussian model. Hence the pixels of the moving foreground objects will demonstrate a relatively high degree of variance. So when a grayish elephant is moving on a brownish ground, it is likely that although there will be a gray color component associated with it, it will deviate from the previous observation by a relatively large margin.

For each pixel, the probability density function decides which of the three Gaussian models represents a closest match. After deciding the closest matching Gaussian model, they compare pixel intensity value to μ to make sure that it falls within 2.5 standard deviation of the μ. If the observed value is within the accepted range, they update the μ and variance σ^2 of the closest matching Gaussian model.

In [2], they propose the use of rho (ρ) every time a pixel is determined to match an existing Gaussian model and replace the use of probability density function as proposed by Stauffer and Grimson, claiming that computation of ρ is more efficient. They use the following formula for updating $\rho = \frac{\alpha}{\omega_k}$, where α is a time varying gain. The other equations for updating mean and variance of the Gaussian model are taken from Stauffer and Grimson [1].

The equation for updating mean is

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t$$

The equation for updating variance is

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

where X_t is the current intensity value of the pixel. For color image it is a vector containing three values, one for each plane in RGB color space, between 0-255.

The weight (**ω**) of each of the Gaussian models is updated according to the formula

$$\omega_{k,t} = (1-\alpha_t)\omega_{k,t-1} + \alpha_t(M_{k,t})$$

Where

$\mathbf{M_{k,t}}$ = 1, for closest matching Gaussian model and
= 0, for other Gaussian models.

α_t = 1/t (is a time varying gain)

But if the observed pixel is not a member of any of the existing three Gaussian models, then a new Gaussian model must be created to represent the currently observed pixel. But since the number of Gaussian models per pixel is limited to three, the newly created Gaussian model must replace one of the existing Gaussian models. They replace the Gaussian model with least weight since it has not been seen very often. This makes the algorithm respond to the persistent scene changes and creates secondary background states for multi-modal backgrounds. Then they initialize the Gaussian distributions of each color plane with the pixel intensity values and set initial variance to a high value and the weight to a low value.

Then based on the typical background characteristics of low variance and high weight, they decide the Gaussian models representing foreground objects. For accomplishing this, they rank all the Gaussian models according to the **ω/σ** and Gaussian models having value of **ω/σ** beyond certain threshold are considered to represent the foreground objects.

This method would not be practical for video taken with panning camera.

2.3 Background

In this work the algorithm used for background subtraction can be used to extract moving deformable foreground objects from the video taken with either static or moving cameras.

For a video taken with a static camera, the frame by frame differencing will give the pixels on the silhouette of the moving objects. But for a panning camera, since the

background is also in a relative motion to the foreground objects, a simple frame by frame differencing will also show pixels on the silhouettes of background objects. Hence the simple frame by frame differencing methods will not work.

For example, the table in the following figure, fig [1], shows the frames 275 to 280, from the video 'car_pan.avi' (taken from [31]) in the 1st column and the difference between two consecutive frames in the 2nd column. Since the camera was panning during these frames, background and foreground were in relative motion. Hence a simple frame differencing shows the pixels on both moving background objects as well as background objects.

Frame 275 	Difference Between275 and276 
Frame276 	Difference Between276 and277 
Frame277 	Difference Between277 and278 

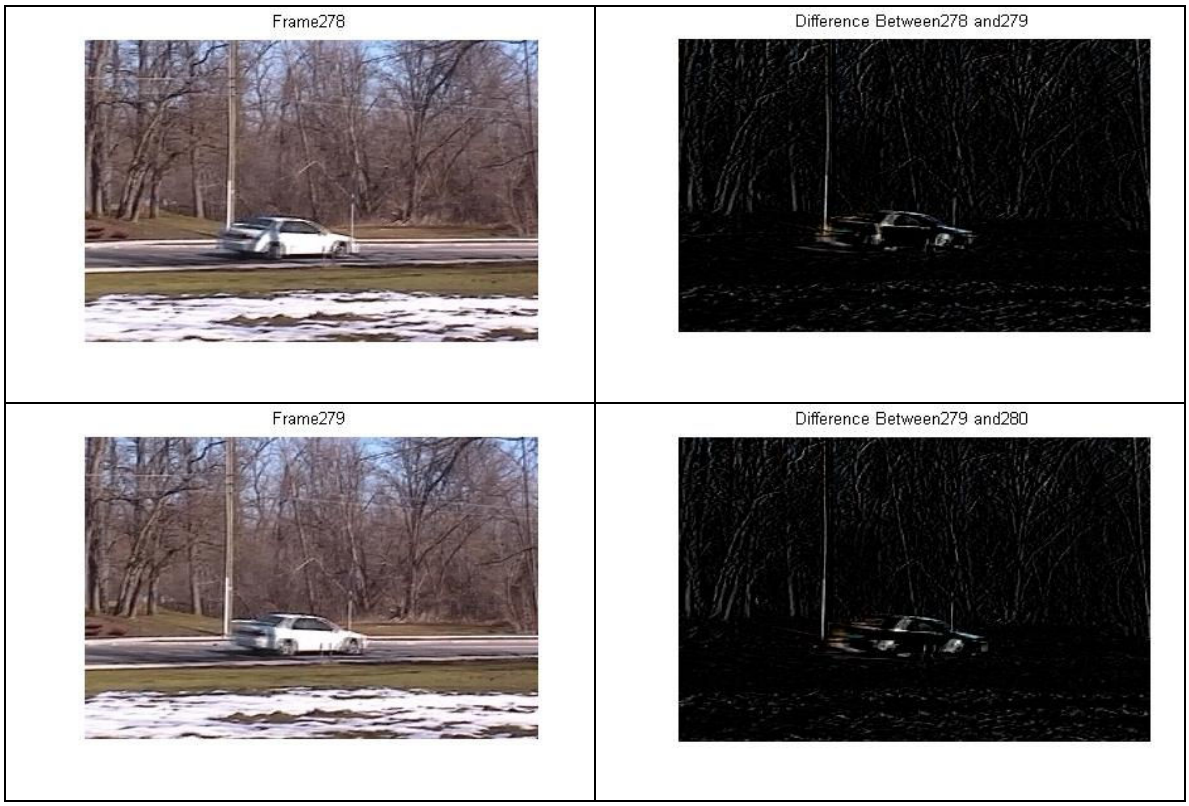


Figure 1. Results showing the effect of the simple frame-by-frame differencing method used on a video taken from a panning camera.

But if the two consecutive frames can be corrected for the motion of the camera, then difference between those two consecutive frames would give the pixels on the silhouette of the moving objects. This is the gist of the algorithm used in this work for extracting the moving foreground objects.

The main task is to correct the two consecutive frames for the motion of the camera. If the magnitude and direction of the motion of the camera can be found, then the current frame, f_t, can be adjusted accordingly and it can be subtracted from the previous frame, f_{t-1}, to give the pixels on the silhouettes of the moving foreground objects.

For a video taken with a panning camera it is reasonable to assume that the camera would not pan fast enough so that the scene in the two consecutive frames is entirely different. Therefore some pixel at location (x, y) in the current frame, f_t, may have its corresponding pixel in the previous frame within the N*N neighborhood of (x, y), where N is the amount of pixels moved by the camera, or if it is located on the scene boundaries then it might have gone out of view. So the current frame, f_t, is just shifted by some amount from the previous frame, f_{t-1}, depending upon the motion of the camera. So correcting the current frame, f_t, for the motion of camera is a problem of image registration.

The detailed algorithm for extraction of foreground object follows the brief explanation of some basic techniques used in the algorithm.

2.3.1 Image Registration

Image registration is a technique of finding point-by-point correspondence between two images of same scene, both spatially and with respect to intensity. A detailed review of different image registration techniques is given in [4].

Image registration is required in many applications like motion analysis, correcting the frames for the motion of the camera, change detection, stereo depth perception, and image fusion.

For a video taken with a panning camera, two consecutive frames of the video are shifted spatially by the amount equal to the motion of the camera. The images of the same scene taken from different angles may be slightly shifted or rotated. So these images, which are either shifted or rotated spatially, can be matched by techniques of image registration by finding the amount by which these images are shifted or rotated and correcting the other image by that much amount. So for the above examples image registration involves finding the optimal spatial transform which can be used to match two or more images. For images taken from different sensors, image registration involves finding optimal intensity transformation. For example, a sensor might take an image in HSV color space and if the application works in RGB color space then intensity transformations can be used for converting them from HSV to RGB color space [4].

2.3.2 Classification of Algorithms for Image Registration

Image registration algorithms can be classified into two main classes: ***area based*** methods and ***feature based*** methods. The original image is commonly known as the reference image and the target image is the image which is to be mapped on the reference image. The algorithms in the ***area based*** methods use a window centered on each pixel and compare that window to the window of same size centered on the pixels in certain neighborhood, in the another image, and then uses some correlation measures like ordinal measure or sum square distances to find degree of similarity between the two windows. Hence they take into account the entire structure of the image. The methods using ***feature based*** techniques, instead of considering entire image structure consider only certain important features of the images such as corners, edges, curves or lines. They match these image features to find the amount by which the two images are spatially misaligned. The features used might be corners, edges, curves or lines. Usually the area based methods are more robust than the feature based methods but are computationally expensive. In this work feature based methods are used [4].

2.3.3 Mapping the images Point-by-Point (Method for Image Registration)

One of the most common methods for mapping the target image(s), having some unknown misalignment, to the reference image is Point-by-Point mapping [4].This method is discussed in detail in [4]. This section summarizes this method and is based on [4]. Three main steps of point-by-point mapping are:

1. ***Feature point extraction:*** Finding interesting feature points in the two images to be registered. In this the work two images to be registered are the two consecutive frames of a video.
2. ***Finding Correspondence:*** Establishing correspondence between the interesting feature points in the images to be registered. In this work, after finding interesting points in the two consecutive frames of a video, a correspondence measure is used to establish correspondence between the interesting points on the two consecutive video frames. After matching the interesting points on the two consecutive frames, the difference between the x and y coordinates for each corresponding interesting points are found. Then the mode of the differences in the x and y coordinates is taken to obtain the net shift in the x and y directions caused due to the motion of the camera.
3. ***Using Appropriate Transformation for Aligning Target Image:*** After finding the net shift in x and y direction caused due to the camera motion, the current frame, f_t, needs to be aligned accordingly using some transformation. After aligning the current frame, it is subtracted from the previous frame, f_{t-1}. Since the current frame is corrected for the motion of camera, the resulting difference image gives the pixels on the silhouettes of the moving foreground objects.

Individual steps of the Point Mapping methods are explained in detail in later sections after giving some overview of Spatial Domain Processing in the following section 2.3.4 and Gaussian Smoothing in section 2.3.5.

2.3.4 Spatial Domain Processing

The methods in Spatial Domain Processing directly manipulate the pixels in the image's intensity plane. The intensity planes of an image (e.g. Red, Green, and Blue intensity planes for an image in RGB color space) are called its spatial domain [9]. There are two main methods in Spatial Domain processing

1. Intensity or Grayscale Transformations.
2. Spatial Filtering.

Both of the above categories of spatial domain processing are discussed in depth in [9]. In this work Spatial Filtering from [9] is explained in brief.

Spatial Filtering is also known as ***Neighborhood Processing*** or ***Spatial Convolution*** [9].

For each pixel P(x, y) in the input image ***f***, an operation is performed on the pixels in the neighborhood of predefined size centered at P(x, y), and the result of that operation is used as a value of the pixel at location (x, y) in the output image ***g***. The neighborhood is typically defined by a square or rectangular window centered at location P(x, y) (location of current pixel). This is also known as ***Spatial Filtering*** [9].

If the computation performed on the pixels in the neighborhood is linear then it is called ***linear spatial filtering*** and if it is non-linear then it is called ***non linear spatial filtering*** [9].

In linear spatial filtering, a sliding window, also called a ***mask***, ***filter, kernel, template, or window,*** is centered around a pixel at location (x, y) in an input image, ***f,*** and generates a new output image, ***g***, by multiplying each pixel value in a certain neighborhood of (x, y) in ***f*** with the corresponding weight in the convolution mask and summing these products. The filter or mask is placed step-by-step over all the pixels in the input image, ***f***, to obtain an output image ***g*** [9].

Linear spatial filtering involves two closely related operations of ***convolution*** and ***correlation***.

In ***full correlation*** a 2D mask '***m'*** is passed by the image '***I'*** in the following way [9]

1. The bottom right most element of the mask is aligned with the top left most pixel of the image.
2. Some points between the functions (mask and the image) do not overlap. To solve this problem the image is padded with as many 0's as necessary.
3. Then the mask is moved in all the possible directions such that atleast one of the pixels of the mask overlaps a pixel in the original image.
4. Then the sum of the products of the mask coefficient and the corresponding pixel value is taken and it is the response of the mask for that particular pixel.

In ***same correlation*** the mask is initially arranged so that its center overlaps the top left pixel of the image and rest of the steps are same.

Process of convolution is almost same as that of correlation except that the mask is shifted by 180 degree [9].

2.3.5 Gaussian Smoothing

When the Gaussian filter is convolved with an image, it removes the noise from the image but also blurs the image and removes the details from it. It is a 'low pass filter'. In working it is similar to an averaging filter. But the mask used in Gaussian smoothing is different from the mask used in an averaging filter. The mask used for Gaussian smoothing represents the shape of Gaussian, which is a bell shaped curve. It removes the noise but also preserves the edges in the image.

The Gaussian filter can be specified by the user by providing the mean and standard deviation.

2D circularly symmetric Gaussian has following form [32]

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

In this work Gaussian filter of size 7*7 and standard deviation of 3 is used for smoothing the frames before finding interesting features in it. This Gaussian filter is shown in the fig [2] and is generated by the following code snippet in Matlab.

```
w = fspecial('gaussian', 7, 3);
surf(w), title('Gaussian Filter used for Smoothing'), xlabel('x'), ylabel('y'), zlabel('z');
```

In OpenCV this filter is generated by using following code snippet

```
cvSmooth(imgSrc, imgDst, CV_GAUSSIAN, 7, 7, 3);
```

Where '***imgSrc'*** is the source image and '***imgDest'*** is the destination image. Both images can have either 1 or 3 channels and can have either 8 bit or 32 bit floating point values.

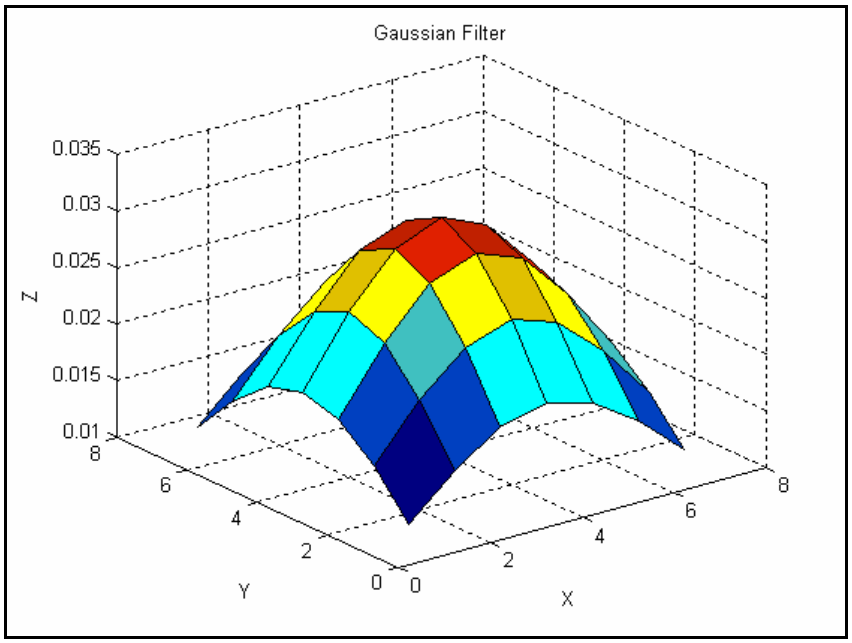


Figure 2. 7*7 Gaussian filter of σ = 3

Applying Gaussian smoothing before edge detection can be useful since it softens the edges and filters out the noise in an image. Using a small Gaussian filter, e.g., of size 3*3 or 5*5 will give fine details but it will also give many unwanted edges due to noise and fine texture. A large smoothing filter may remove most of the unwanted edges but much of the details might also be lost [13].

Figure 3. Original Image 'CIS_Building.jpg'. Taken from [31].

For example, a noise is added to the image in fig [3] using the Matlab function '***imnoise'***. The type of noise added is 'Salt & Pepper' with density 0.02. The image after adding noise to it looks like fig [4].

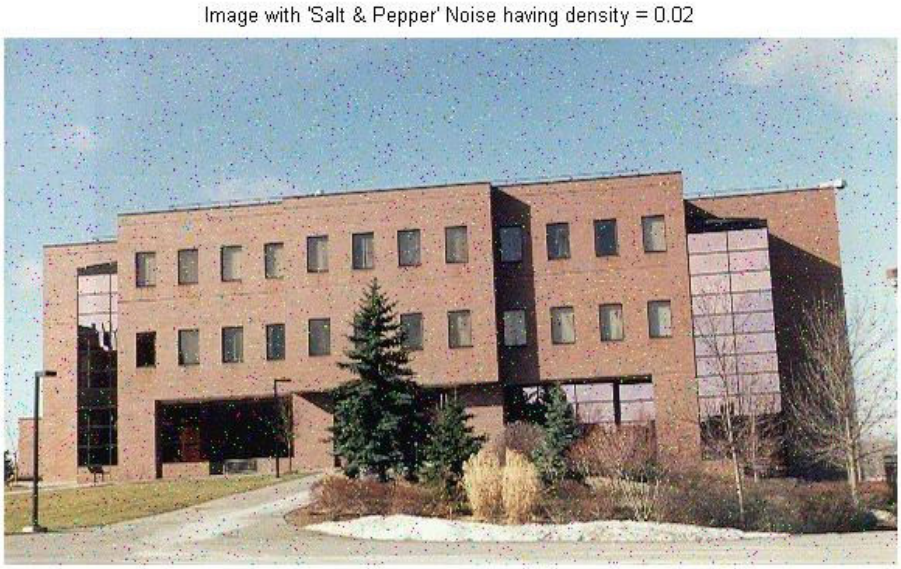


Figure 4. After adding 'Salt & Pepper' noise with density = 0.02 to fig [3].

To get rid off the noise the image is smoothed with the Gaussian filter, shown in fig [2], and the resulting image is fig [5].

Figure 5. After filtering the image with the Gaussian Filter most of the noise is removed but the image is also blurred.

2.3.6 Feature Point Extraction

In this work, the video can be taken either from a moving camera or a stationary camera. For a video taken with a panning camera two consecutive frames of that video can give the information about the relative velocity of the camera. The amount by which each pixel in the current frame is shifted from the previous one can be computed and the mode of the shift values in the x and y directions of all the pixels in current frame can be computed to obtain the shift introduced by the movement of the camera. But comparing every pixel in the current frame with all the pixels within certain neighborhood of the previous frame is computationally very expensive. Hence in this work only feature points (corners) found in the current frame are mapped to their corresponding feature points in the previous frame and the shift in their spatial coordinates is found and the current frame is adjusted for that amount of the shift. This reduces the time complexity of the algorithm [29].

Feature point extraction involves finding interesting points in an image. In this work interesting points are found on the previous frame, $\boldsymbol{f_{t-1}}$, and the current frame, $\boldsymbol{f_t}$.

Any typical indoor or outdoor video taken from either a stationary or moving camera will have objects on background and moving foreground objects. For e.g. a car moving on the road contains the moving car as the foreground objects and road or trees or houses in the background. So for a video taken with a panning camera both the background and foreground objects will be in relative motion. After finding the amount and direction of the shift in the static background objects the relative velocity of the moving camera can be estimated. Hence by focusing only on the objects in the video the shift in the camera can be obtained. Objects in an image can be highlighted by finding the edges in the image since edges represents boundaries of the object in the image. For the objects having different color than the background, the edge corresponds to the boundary between the object and the background or different part of the same object. Hence an edge indicates a sudden jump in intensity from one pixel to another pixel in the direction perpendicular to the edge. Hence an edge indicates a sharp contrast in the intensity [14].

Detecting edges in an image is most common approach of finding meaningful discontinuities in an image rather than detecting points and lines in an image. To detect edges in an image, discontinuity in intensity values need to be detected by the first order and the second order derivatives of the image, as derivatives gives rate of change of the some quantity.

Derivatives of an image are explained in detail in [15]. The following paragraphs summarize the work in [15] and the formulas are taken from [15].

Derivatives are required to find the sudden changes in the image intensity indicating edges or corners. But OpenCV or Matlab treats an image like a 2D or 3D array of pixels, i.e., a grayscale image $\boldsymbol{I}$ of size *width*height* is an array $\boldsymbol{I}$ of *width* columns and *height* rows, with I(x, y) giving the intensity of the pixel at the location (x, y). Hence the image is a discrete function of variables x and y which are its spatial coordinates. But the

derivatives can be found only for the continuous functions. Hence for the purpose of derivations, image is considered to be sampling of a continuous function of the variables x and y [15].

A vertical edge in the image indicates a sharp contrast in the intensity in the horizontal direction i.e. the direction perpendicular to it. Vertical edges in the image can be found by taking the derivative of the image in the horizontal direction. Hence while taking the derivative of the image with respect to x axis i.e. the horizontal axis any derivative with a very high positive or negative value indicates a vertical edge at that x coordinate [15].

For a continuous function *f(x, y)* its partial derivative with respect to *x* axis is given by

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

where Δx is an arbitrarily small value [15]. But if *f(x, y)* is a discrete function representing a 2D digital image then the value of Δx is 1 pixel which is the smallest in the image. Hence the derivative of the image at location (i, j) is given by [15]

$$\frac{\partial f(x,y)}{\partial x} \approx f(i, j+1) - f(i, j)$$

Where j=x and i=y.

This is same as convolving the image with the mask of [1 -1].

Hence the first derivative of image can be found by convolving it with the following masks

1. [1 -1] to find derivative with respect to horizontal i.e. ***x*** direction.
2. $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ to find derivative with respect to vertical i.e. ***y*** direction

The other variant of the filters used are

1. [1 0 -1] to find derivative with respect to horizontal i.e. ***x*** direction.
2. $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ to find derivative with respect to vertical i.e. ***y*** direction

Similarly the 2nd order derivative of the image can be found by convolving the image with following filters or masks

1. [1 -2 1] to find derivative with respect to horizontal i.e. ***x*** direction.

2. $\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ to find derivative with respect to vertical i.e. ***y*** direction

Since and edge indicates the change in intensity its 1st and 2nd derivative indicates the amount and direction of this change.

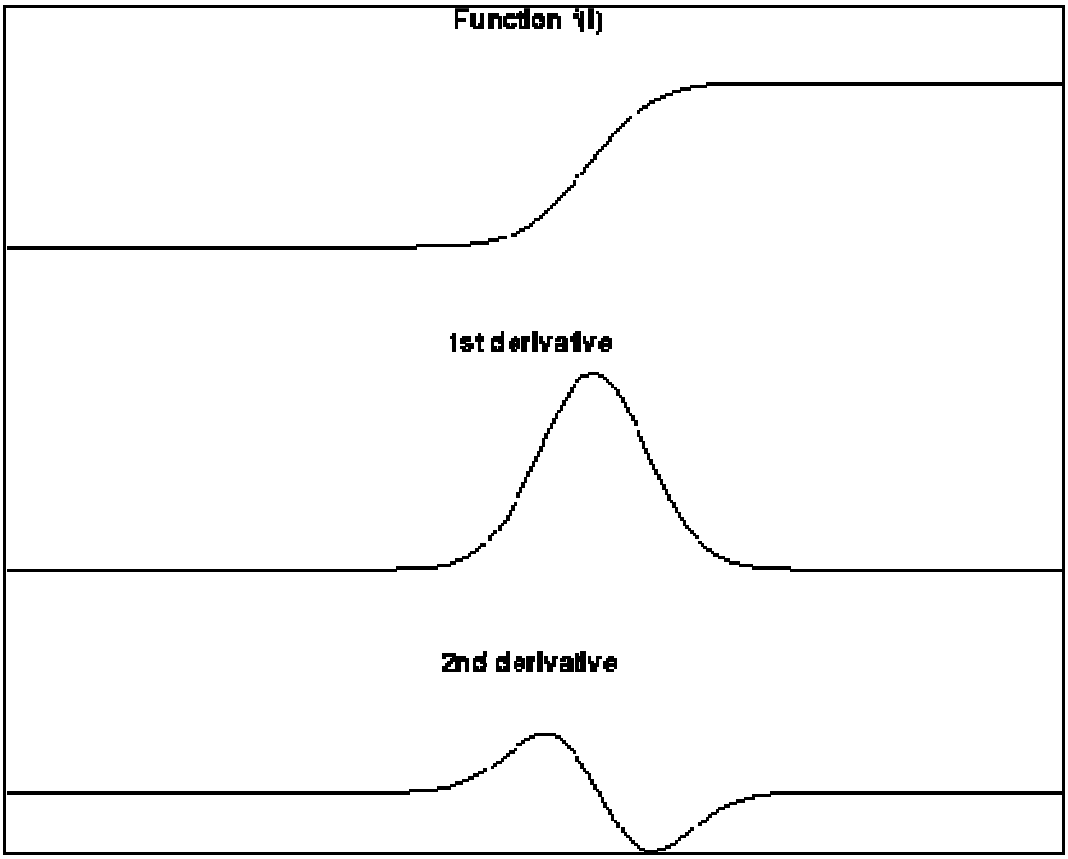


Figure 6. Figure indicating the 1st and 2nd derivative of an edge. Taken from [10].

The two most common approaches to find the 1st and 2nd derivatives of the image are ***Prewitt compass edge detection*** and ***gradient edge detection***.

In Prewitt edge detection the image is convolved by a set of eight masks, each of which is sensitive to a different edge orientation. The mask which produces the maximum response determines the magnitude and direction of the edge. Different types of masks are Sobel, Prewitt, Roberts etc.

Gradient edge detection is the most common method for detecting edges in the image. The image is convolved with only 2 masks, one finding the gradient in x-direction and other finding the gradient in y-direction.

The concept of edge detection using gradient is explained in detail in [9]. The following formulas for gradient are taken from [9].

For a 2D function, f(x, y), its gradient is defined as the vector [9]

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude of this vector is given by [9]

$$|\nabla f| = \left[G_x^2 + G_y^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]$$

It is often approximated by [9]

$$|\nabla f| \approx |G_x| + |G_y|$$

"The gradient vector points in the direction of the maximum rate of change of the function *f* at coordinates (x, y) and the angle at which this maximum rate of change occurs is given by" [9]

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

The horizontal gradient, G_x, at a pixel can be found by convolving it with the mask [-1 0 1] and the vertical gradient at a pixel can be found by convolving it with the mask $[1\ 0\ -1]^T$.

In this work corners are used as interesting points. It is assumed that most of the corners found in the previous frame, $\boldsymbol{f_{t-1}}$, would be preserved in the current frame, $\boldsymbol{f_t}$. Few of the corners found in the previous frame might be out of view in the current frame due to the motion of the camera. But most of the corners found in the previous frame will be still found in the current frame, but they might be shifted by certain amount depending upon the motion of the camera.

Before finding the corners the image is usually smoothed with the ***Gaussian filter*** to remove any noise so as to reduce the false corner detection.

The corners detected on the noisy image in fig [4] are shown below in fig [7].

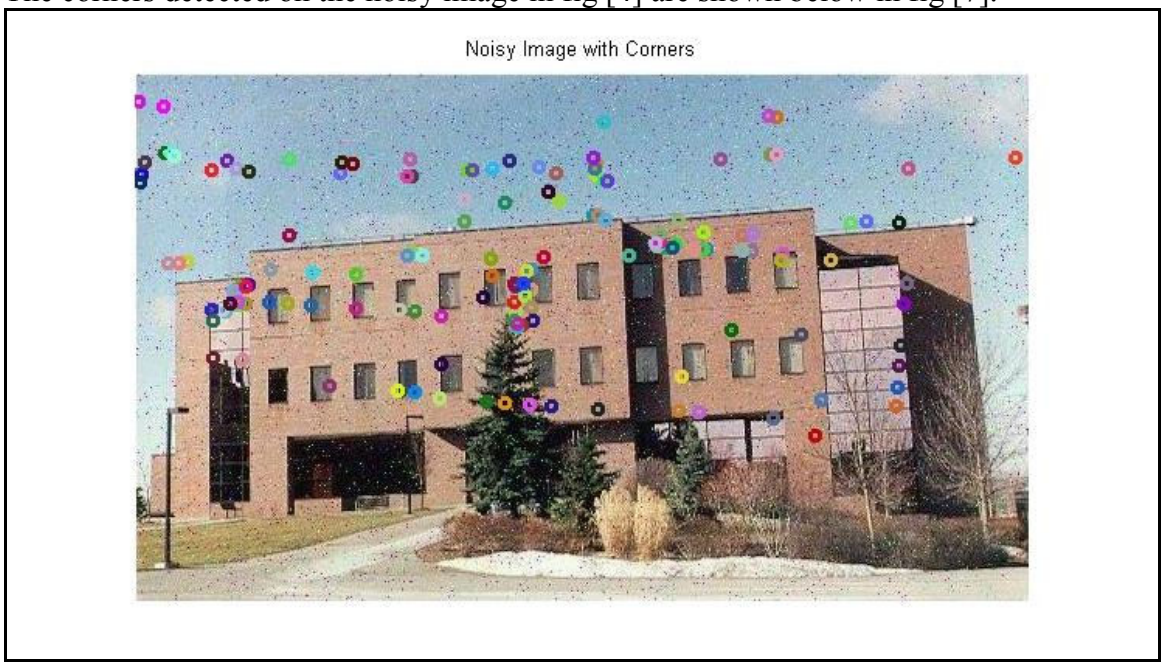


Figure 7. Corners detected on the image in fig [4].

The corners detected on the image, which is obtained after smoothing the noisy image (fig [5]), are shown in fig [8].

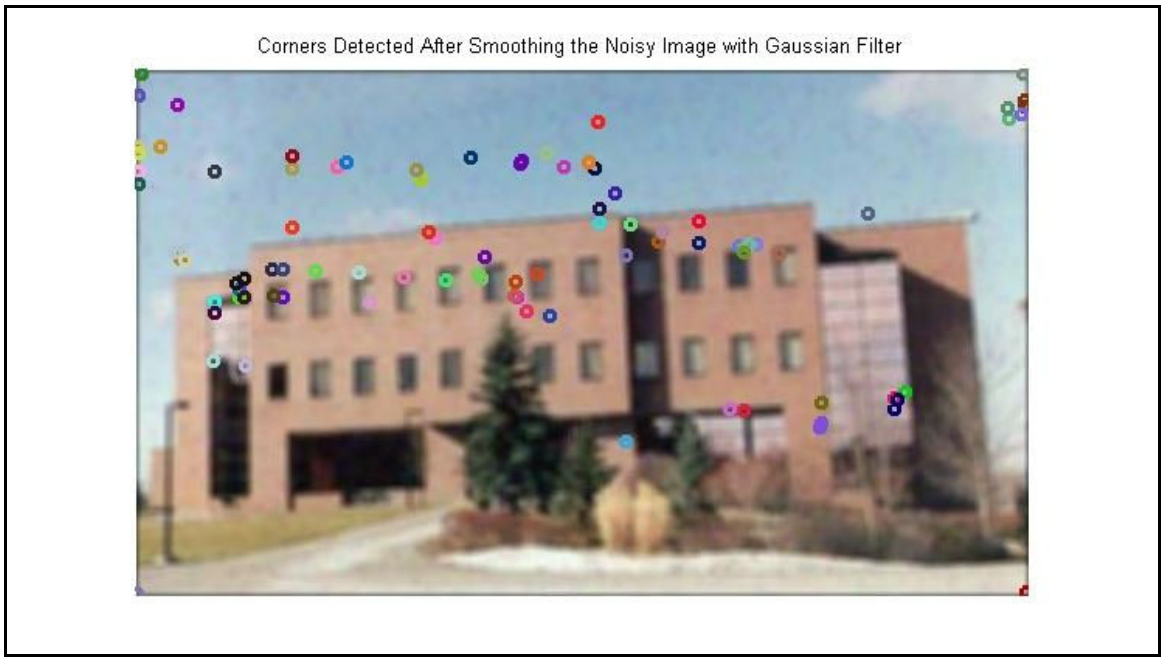


Figure 8. Corners detected on the image obtained in fig [5].

The number of false corners detected, after smoothing the image with Gaussian filter, is much fewer as compared to the false corner detected in the noisy image without smoothing. For example, in fig [8] the corners in the trees, glass window wall on right side and sky are much fewer as compared to the corners in fig [7].

But since smoothing causes blurring of the image, the number of the true corners detected after smoothing the image also decreases. For example, there are fewer true corners around the window glasses.

But in this work, the frames are smoothed before finding the corners because the reduction in the number of the true corners caused after smoothing the frame isn't drastic. There are still enough true corners to do frame registration. Also after smoothing, the false corners caused due to noise are removed and hence the chance of getting better registration increases.

There are many corner detectors. But this work uses ***Harris Corner Detector***. Since the corners are formed by intersection of two or more edges and edges usually define the boundary between two different objects or parts of same object, they are interesting.

A detailed comparison of all the corner detectors can be found in [29]. This work summarizes most common corner detectors and is based on [29].

Most of the corner detectors have the same general algorithm having following steps

1. ***Applying Corner Operator to Obtain Cornerness map:*** All the corner detectors have some type of corner operator which assigns a cornerness value to every pixel in an image. Cornerness value is measure which indicates how well a pixel represents a corner. The corner operator is a main differentiating factor between different corner detectors. A cornerness map gives the cornerness measure of every pixel in the input image and hence has the same dimension as that of the input image.

2. ***Thresholding Cornerness Map:*** From the Cornerness map obtained in step 1, pixels with the local maximum cornerness values have chances to represent a corner. But there might be many pixels which are local maximum but their cornerness value is too small to represent a corner. Hence a threshold is required to avoid detection of local maximum pixels with low cornerness values as corners. Threshold value is application dependent.

3. ***Finding Local-Maximum:*** After thresholding the cornerness map in step 2, the local maximum values in cornerness map are found and are declared as corners. For each point in cornerness map its certain n*n neighborhood is observed and if the cornerness value of that point is less than the cornerness value of any other point in its n*n neighborhood, then its cornerness value is set to 0.

The most common corner detectors are

1. ***Moravec Corner Detector:*** Moravec operator defines a point having large intensity variations in all the eight principle directions (left, right, top, bottom, top-left, top-right, bottom-right and bottom-left) as an interesting point. Since this property is similar to that of the corner, Moravec operator is considered to be the corner detector.

   For every pixel I(x, y) at location (x, y) in an image, Moravec Operator finds intensity variation at that pixel in by

   1. Placing a square window (typically of size 3*3, 5*5 or 7*7) centered at (x, y) and shifting that window by one pixel in each of the eight principle directions.
   2. For a particular shift intensity variation is calculated as the sum of the square of the intensity differences between the corresponding pixels in these two windows (shifted window and the original window centered at (x, y)).
   3. After finding the intensity variation for each of the eight shifts the minimum intensity variation is selected as the intensity variation at location (x, y).

   Mathematical notations of Moravec Operator can be given by

$$E_{x,y} = \sum_{u,v} w_{u,v} \left| I_{x+u,y+v} - I_{u,v} \right|^2$$

   Where ***w*** is the image window having the value 1 in the specified rectangular region and 0 elsewhere. x and y are the shifts in 8 principle directions having value (1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (1, -1), (-1, 1) and (-1, -1).

   There are four cases that needs to be considered while using Moravec Operator

   1. If the portion of the image under the window is of constant intensity, then all the shifts will result in negligible intensity change.
   2. If the window is along the edge, then any shift along the edge results in small change but any shift perpendicular to the edge results in significant changes.
   3. If the window is on the isolated point then all the shifts will result in large change.
   4. If the window is on the corner, then all the shifts will result in the large change. So if the minimum of all the intensity variations, produced by the shifts in eight principle directions, is greater than the threshold then the corner is detected.

The Moravec Corner detector has following drawbacks

1. Since the shifts in only eight principle directions, i.e., at every 45 degrees is considered the response of the operator is very anisotropic.
2. Since the window considered is rectangular and binary the response is noisy.
3. Since the minimum intensity variation out of all the shifts is taken into account, it detects corners all along the diagonal edges which don't lie parallel to the eight principle directions. Hence the Moravec operator is not rotation invariant. If the image is rotated and after rotation most of its edges are not parallel to the eight principle directions then the number of corners found will be too large.

A detailed explanation of Moravec operator and its drawbacks is given in [29].

2. ***Harris/Plessey Corner Detector:*** Harris corner detector overcomes many drawbacks of the Moravec Corner detector.

   1. Moravec corner detector was anisotropic since it considered shifts at every 45 degrees. Harris corner detector removes this drawback by performing intensity variation measurement at all possible small shifts. The change E produced by any arbitrary shift (x, y) is given by

$$E_{x,y} = \sum_{u,v} w_{u,v} \left[x \frac{\partial I}{\partial x} + y \frac{\partial I}{\partial y} \right]^2$$

   Where $\frac{\partial I}{\partial x}$ = I $\otimes$ [-1 0 1] and $\frac{\partial I}{\partial y}$ = I $\otimes$ [-1 0 1]^T and $\otimes$ is convolution operator.

   For (x,y) having values (1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (1, -1), (-1, 1) and (-1, -1) it behaves approximately like Moravec Corner Detector.

   This new measure of intensity variation can measure intensity changes in all possible directions by proper selection of x and y values.

   The x and y values are the distances in horizontal and vertical direction required to construct a line in a unit triangle in the direction of intensity change **"29"**. For example finding the intensity variation in the direction of 45 degrees the value of x and y is 0.5.

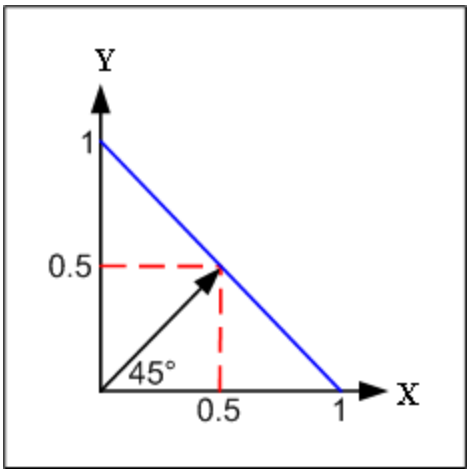


Figure 9. Method of computing value of x and y. Taken from [29]

2. Moravec Operator uses rectangular and binary window. Since the window is square all the pixels in corner of the square are not equidistant from the center. Harris corner detector uses a circular window so the Euclidean distance from the center pixel of the window to the edge of the window is same in all the directions and hence gives better measurement of the local intensity variation. Also window in Moravec operator is binary, i.e., it assigns equal weights to all the pixels in the window. But the pixels near the center of the window are better indicator of the local intensity variations as compared to the pixels on the edges of the window. So pixels near the center of the window should be given the higher weight.

   Hence the Harris corner detector uses a circular Gaussian window given by

$$w_{u,v} = e^{\frac{-\left(u^2+v^2\right)}{2\sigma^2}}$$

3. Moravec operator detects many corners along the edges which are not parallel to any of the eight principle directions.

   "Any imperfections in an edge due to intensity quantization, pixilation or noise may result in the increase of the minimum local intensity variation". [29]. Harris corner detector overcomes this drawback by reformulating it in the following way (Derivation taken from [29]).

$$E_{x,y} = \sum_{u,v} w_{u,v} \left[x \frac{\partial I}{\partial x} + y \frac{\partial I}{\partial y} \right]^2$$

$$= \sum_{u,v} w_{u,v} \left[x^2 \left(\frac{\partial I}{\partial x} \right)^2 + 2xy \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} + y^2 \left(\frac{\partial I}{\partial Y} \right)^2 \right]$$

$$= Ax^2 + 2Cxy + By^2$$

Where A = $\left(\frac{\partial I}{\partial x} \right)^2 \otimes w$, B = $\left(\frac{\partial I}{\partial y} \right)^2 \otimes w$ and C = $\left(\frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \right) \otimes w$

They further represent the equation as

$$E_{x,y} = Ax^2 + 2Cxy + By^2$$

$$= \begin{bmatrix} x & y \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix}$$

Where $M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$ [29].

Reformulated cornerness measure considers the variation between the intensity changes in different direction.

If the 2D image is visualized as a 3D surface where x and y coordinates of the image are the two dimensions and the 3rd dimension is the pixel intensity, then the matrix *M* contains all the differential operators describing the image surface at a given point (x, y).

"Suppose λ_1 and $\lambda 2$ are the eigenvalues of *M*, then they will be proportional to the principle curvature of the image surface and form a rotationally invariant description of *M*" [29].

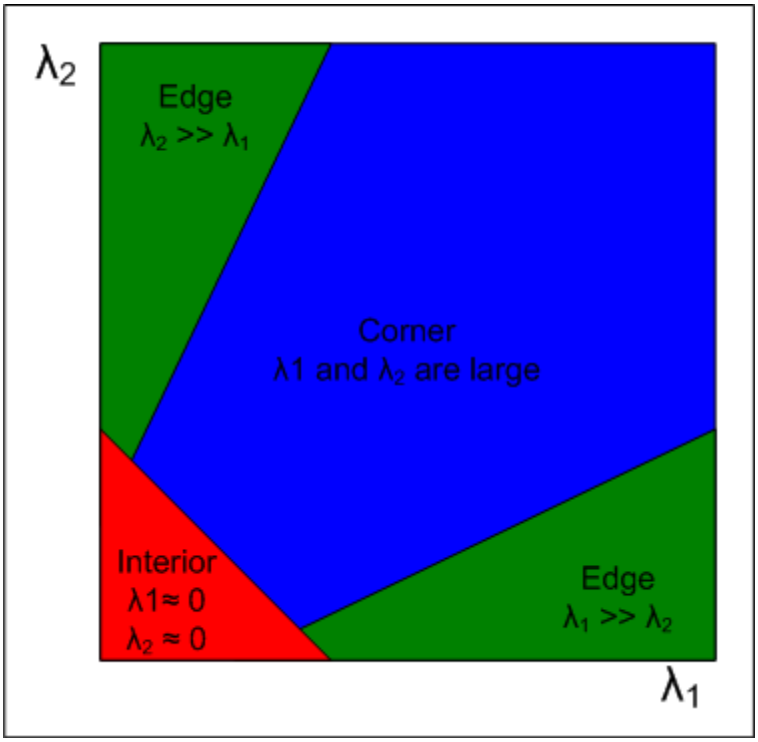


Figure 10. Eigenvalues indicating the principle curvature of the image surface. Taken from [29].

There are three different cases to be considered

1. For the portion of the image within the window having relatively constant intensity, there is little curvature and hence both the eigenvalues are relatively small, which indicates there is no corner.
2. For the window along the edge, there is a very less curvature along the edge and a very high curvature perpendicular to the edge i.e. one curvature is high and another curvature is low forming a ridge indicating the edge.
3. For corner and isolated point both the curvature will be high so both eigenvalues would be high.

To obtain the cornerness map, Harris corner detector uses the following cornerness measures [29]

$$C(x,y) = \det(M) - k(trace(M))^2$$
$$\det(M) = \lambda_1\lambda_2 = AB - C^2$$
$$trace(M) = \lambda_1 + \lambda_2 = A + B$$

Where k = constant.

Algorithm for Harris Corner Detector

1. For each pixel at coordinate (x, y) in the image compute the autocorrelation matrix *M*.
2. Compute the cornerness measure *C(x, y)* and create the cornerness map.
3. Set all the points in cornerness map having cornerness measure less than the threshold to 0, i.e., if C(x,y) < threshold then C(x, y) = 0.
4. Use non-maximal suppression to find local maxima.

In this work, Gaussian smoothing was performed before using Harris Corner Detector to get rid of noises.

In this work, OpenCV [30] function ***cvGoodFeaturesToTrack()*** is used to determine strong corners in image. ***cvGoodFeaturesToTrack()*** has a parameter which if set to non-zero uses Harris Corner Detector to detect corner and if set to zero then finds corner with large eigen values in an image.

Below are the two consecutive frame, frame 260 and frame 261, which are taken from a video ***'car_pan.avi'***. The video was taken from [31].

Figure 11. Frame 260 of video 'car_pan.avi'. Taken from [31].

Figure 12. Frame 261 of video 'car_pan.avi'

After finding corners in the above two frames the resulting frames are shown in following figures, fig[13] and fig[14].

Figure 13. Corners found in Frame 260

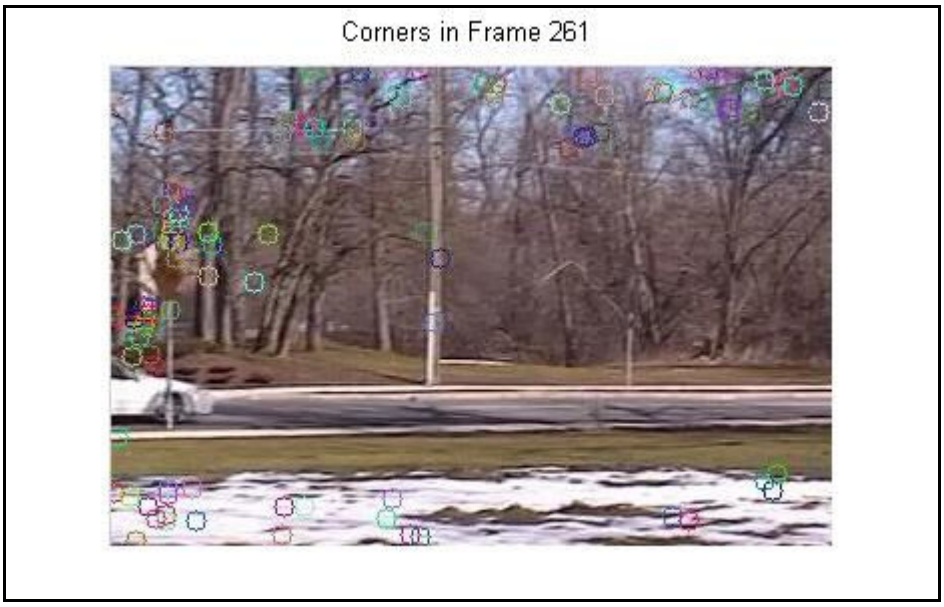


Figure 14. Corners found in Frame 261.

2.3.7 Correspondence Measures

After finding the interesting points, i.e., the corners in two consecutive frames, all the feature points in the previous frame, f_{t-1}, must be matched with their corresponding feature points in the current frame, f_t. The general method of finding correspondence is, compare every feature point, P(x, y), in the previous frame, f_{t-1}, with all feature points in the current frame, f_t, in certain $N*N$ neighborhood of location (x, y), where N is the maximum amount of shift which can be caused by camera in two consecutive frame. In this work the value of N is assumed to be 20, i.e., it is assumed that camera wouldn't pan fast enough to cause a shift in pixel in two consecutive frames by amount more than 20 in any given direction.

To compare a feature point P1(x, y) (pixel at location (x, y)) in the frame f_{t-1} with a feature point P2(x', y') in the frame f_t, usually their intensity values are not compared directly because it can be error prone due to noise effects. Instead a square window, ***w1***, of certain size (for e.g. 3*3, 5*5 or 7*7 etc) centered at P1(x, y) and another square window, ***w2,*** of same size centered around P2(x', y') are compared. The distance between these square windows, ***w1***and ***w2,*** is found. This distance is measure of how close these two pixels P1(x, y) and P2(x',y') are.

The most common method of finding the distance between these two windows is sum of squared difference between corresponding intensities.

For example, if the windows of size 3*3 centered around P1(x, y) (A5) and P2(x', y') (B5) are

Intensity Matrix (I_1)

55 A1	31 A2	44 A3
53 A4	**37** **A5**	45 A6
45 A7	38 A8	51 A9

Intensity Matrix (I_2)

55 B1	31 B2	44 B3
53 B4	**37** **B5**	45 B6
45 B7	38 B8	51 B9

The Sum Squared Distance in given by

$$\mathbf{SSD} = \sum_{i=1}^{9} (A_i - B_i)^2$$

But in this work the correspondence measure used to match the feature points in two consecutive frames is taken from [8]. It is called ***Ordinal Measures.***

According to the Ordinal Measures, to compare two corners for similarity, instead of directly comparing their N*N neighborhood of the intensity plane, the pixels in their N*N neighborhoods are ranked according to their intensity to get a ***rank matrix*** of dimension N*N and instead of the N*N intensity matrices those N*N rank matrices are compared. They also propose a measure to find the distance between the two rank matrices. This makes the correspondence measure invariant to changes in illumination caused by shadows, sun light or any light in indoor scenes.

It is based on the idea that any change in intensity caused due to shadows, sun light or any other indoor light will probably effect not only the pixel but also its small N*N neighborhood equally. So the pixels in the N*N window should preserve their ranks. Hence, instead of comparing the intensity matrices directly, the intensity matrices are converted to rank matrices and the rank matrices are compared.

For example, the intensity matrix and its corresponding rank matrix for the pixels in 3*3 neighborhood of a corner are

Intensity Matrix

17	33	75
24	52	87
41	68	94

Rank Matrix

1	3	7
2	5	8
4	6	9

They also define a measure to find distance between the two rank matrices, i.e., how close the 2 rank matrices are. Let $\boldsymbol{I_1}$ and $\boldsymbol{I_2}$ be two ***N*N*** intensity matrices. The corresponding ***N*N*** rank matrices are π_1 and π_2. They define a ***composition permutation***, ***S***, as follows

$$S^i = \pi_2^k$$

$$\mathbf{k} = (\pi_1^{-1})^{\boldsymbol{i}}$$

Where ***i*** = 1 to ***N*N***. π_1^{-1} denotes the inverse permutation of π_1. They define the inverse permutation as: if π_1^i = j, then $(\pi_1^{-1})^j$ = i. Hence $\mathbf{S^i}$ is the rank of the pixel in $\boldsymbol{I_2}$ that corresponds to the pixel with rank ***i*** in $\boldsymbol{I_1}$. Under perfect correlation, ***S,*** should be identical to the identity permutation given by ***U*** = {1, 2, 3… ***n*n***}**.**

Then they define a distance measure between ***S*** and ***U*** which in turn gives the distance measure between π_1 and π_2. They define deviation d_m^i at each $\mathbf{S^i}$ as the number of $\mathbf{S^j}$, ***j*** = 1,…,***i*** greater than ***i***

$$d_m^i = \sum_{j=1}^{i} J(S^j > i)$$

$$= i - \sum_{j=1}^{i} J(S^j \leq i)$$

Where J(B) = 1 when B is true and 0 otherwise.

The vector d_m^i is termed as distance vector $\boldsymbol{d_m\ (s, u)}$. It estimates the number of ***S*** elements that are out of position. If $\boldsymbol{I_1}$ and $\boldsymbol{I_2}$ were perfectly correlated, then $\boldsymbol{d_m\ (s, u)}$ = (0, 0, …., 0). The maximum value that any component of the distance vector can take is $\left\lfloor \frac{n}{2} \right\rfloor$ which occurs in the case of perfect negative correlation.

Then they define a measure of correlation $\kappa = \kappa(\boldsymbol{I_1}, \boldsymbol{I_2})$ as

$$\kappa(\boldsymbol{I_1}, \boldsymbol{I_2}) = 1 - \frac{2 \max_{i=1}^{n} d_m^i}{\left\lfloor \frac{n}{2} \right\rfloor}$$

If $\boldsymbol{I_1}$ and $\boldsymbol{I_2}$ were perfectly correlated then κ=1 and κ=-1 when they are perfectly uncorrelated.

For example, for a pixel intensity matrix $\boldsymbol{I_1}$ in the frame $\boldsymbol{f_{t-1}}$ and intensity matrix $\boldsymbol{I_2}$ in the frame $\boldsymbol{f_t}$ and their corresponding rank matrices π_1 and π_2 are as follows

I_1

29	38	47
61	42	58
39	71	44

I_2

33	42	44
59	47	62
68	37	54

π_1

1	2	6
8	4	7
3	9	5

π_2

1	3	4
7	5	8
9	2	6

Therefore $\pi_1 = \{1, 2, 6, 8, 4, 7, 3, 9, 5\}$ and $\pi_2 = \{1, 3, 4, 7, 5, 8, 9, 2, 6\}$

$\pi_1^1 = 1,\ \pi_1^{-1} = 1,\ \pi_2^1 = 1,\ S^1 = 1$
$\pi_1^2 = 2,\ \pi_1^{-2} = 2,\ \pi_2^2 = 3,\ S^2 = 3$
$\pi_1^7 = 3,\ \pi_1^{-3} = 7,\ \pi_2^7 = 9,\ S^3 = 9$
$\pi_1^5 = 4,\ \pi_1^{-4} = 5,\ \pi_2^5 = 5,\ S^4 = 5$
$\pi_1^9 = 5,\ \pi_1^{-5} = 9,\ \pi_2^9 = 6,\ S^5 = 6$
$\pi_1^3 = 6,\ \pi_1^{-6} = 3,\ \pi_2^3 = 4,\ S^6 = 4$
$\pi_1^6 = 7,\ \pi_1^{-7} = 6,\ \pi_2^6 = 8,\ S^7 = 8$
$\pi_1^4 = 8,\ \pi_1^{-8} = 4,\ \pi_2^4 = 7,\ S^8 = 7$
$\pi_1^8 = 9,\ \pi_1^{-9} = 8,\ \pi_2^8 = 2,\ S^9 = 2$

S = {1, 3, 9, 5, 6, 4, 8, 7, 2}

$d_m^1 = 1 - J(S^1 \le 1) = 1 - 1 = \mathbf{0}$
$d_m^2 = 2 - [\, J(S^1 \le 2) + J(S^2 \le 2)] = 2 - [1+0] = \mathbf{1}$
$d_m^3 = 3 - [\, J(S^1 \le 3) + J(S^2 \le 3) + J(S^3 \le 3)] = 3 - [1+1+0] = \mathbf{1}$
$d_m^4 = 4 - [\, J(S^1 \le 4) + J(S^2 \le 4) + J(S^3 \le 4) + J(S^4 \le 4)] = 4 - [1+1+0+0] = \mathbf{2}$
$d_m^5 = 5 - [\, J(S^1 \le 5) + J(S^2 \le 5) + J(S^3 \le 5) + J(S^4 \le 5) + J(S^5 \le 5)]$
$= 5 - [1+1+0+1+0] = \mathbf{2}$
$d_m^6 = 6 - [\, J(S^1 \le 6) + J(S^2 \le 6) + J(S^3 \le 6) + J(S^4 \le 6) + J(S^5 \le 6) + J(S^6 \le 6)]$
$= 6 - [1+1+0+1+1+1] = \mathbf{1}$
$d_m^7 = 7 - [\, J(S^1 \le 7) + J(S^2 \le 7) + J(S^3 \le 7) + J(S^4 \le 7) + J(S^5 \le 7) + J(S^6 \le 7) +$
$J(S^7 \le 7)] = 7 - [1+1+0+1+1+1+0] = \mathbf{2}$
$d_m^8 = 8 - [\, J(S^1 \le 8) + J(S^2 \le 8) + J(S^3 \le 8) + J(S^4 \le 8) + J(S^5 \le 8) + J(S^6 \le 8) +$
$J(S^7 \le 8)] = 8 - [1+1+0+1+1+1+1+1] = \mathbf{1}$
$d_m^9 = 9 - [\, J(S^1 \le 9) + J(S^2 \le 9) + J(S^3 \le 9) + J(S^4 \le 9) + J(S^5 \le 9) + J(S^6 \le 9) +$
$J(S^7 \le 9)] = 9 - [1+1+1+1+1+1+1+1+1] = \mathbf{0}$

$$d_m = \{0, 1, 1, 2, 2, 1, 2, 1, 0\}$$

$$K(I_1, I_2) = 1 - \frac{2*2}{\left\lfloor \frac{9}{2} \right\rfloor} = 0$$

2.3.8 Transformations

To summarize, in the previous sections interesting points are found in the current frame and the previous frame and after that they are matched according to ***ordinal measures***. After matching the feature points the amount by which each of them are spatially shifted is found and the mode of all the shifts gives the amount and direction of the shift introduced by the moving camera. To find the pixels on the silhouettes of moving foreground objects, the current frame needs to be shifted in the direction and by the amount of the shift introduced by the panning camera. Hence a proper transformation is needed to align the current frame according to the previous frame.

In this work the source of misregistration is the panning camera and hence a proper transformation should be selected to remove the spatial distortions.

The main transformations are rigid, affine, projective, perspective and polynomial.

The ***rigid transformation*** can be used for registering the images having distortions introduced either due to translation or reflection. It maintains the relative shape and size of the objects in the reference and target images. It involves recognition of same objects in the reference and target images and finding their coordinates. After finding this information cross-correlation is applied and then through some image manipulation input images are either overlaid or expanded to get a larger output images.

Affine transformations maintain collinearity, i.e., all points lying on a line initially still lay on a line after transformation, and the ratios of distances, for example, midpoint of line segment remains the mid point after transformation. Affine transforms are most commonly used transformation since they are sufficient to match two images taken from same viewing angle but different position. Affine transformation consists of the Cartesian operations of a scaling, a translation and a rotation. It does not change the overall geometric relationships between the points. It does not necessarily preserve angles or lengths [6].

An affine transform has typically four parameters t_x, t_y, s and θ which map a point (x_1, y_1) of the first image to a point (x_2, y_2) of the second image according to the following formula (equation taken from [4])

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s * \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

For registration of the images involving other spatial distortions like skew and aspect rations the general 2D affine transform used is (equations taken from [4])

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix} + \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

The ***perspective transformations*** are used when the images to be registered has distortions introduced due to projection of 3D scene onto a 2D plane. Due to projective distortions the objects which are father away from the camera appear smaller and the objects which are more inclined away from the camera appear to be more compressed [4]. The mapping between the coordinates of object in 3D world, (x, y, z), and the corresponding coordinates (x', y'), in 2D image is given by (Equations taken from [4])

$$x' = \frac{-fx}{z-f}$$

$$y' = \frac{-fy}{z-f}$$

where f is the focal length of the camera.

The ***rectification transformation*** is used in registering a scene tilted with respect to image plane to a tilt free target image of desired scale.

Complicated differences between reference and target images like different positions, different angels and shearing can be handled by affine transformation by mapping straight lines from two separate images. But the moving foreground object extraction algorithm in this work takes into account only the translation, because for any two consecutive frames shearing and rotation doesn't have significant effect and implementing it would be computationally expensive. Simple translation is used to correct the current frame after interesting features in current and previous frames have been matched.

2.4 Algorithm used for Background Subtraction in this Work

For finding correspondence between two consecutive frames, f_{t-1} and f_t, first the corners are found in the frames f_{t-1} and f_t and then for every corner, (x, y), in f_{t-1} its corresponding corner, (x', y'), in the frame f_t is found by comparing the 11*11 rank matrix of the corner, (x,y), in f_{t-1} with 11*11 rank matrix of every corner in 41*41 neighborhood of (x, y) in f_t, i.e., rank matrix of all the corners in frame f_t in the range (x-20 to x+20, y-20 to y+20) are compared with rank matrix of the corner (x, y) in frame f_{t-1} and the most matching point is declared as the corresponding point of corner (x, y).

After matching a corner point P1(x, y) in f_{t-1} with corner point P1(x', y') in f_t the amount by which camera pans is obtained by taking difference between x and y coordinates, i.e., for the point P1(x, y) the shift in x direction is (x'-x) and the shift in y direction is (y'-y).

This shift is found for all the corners in f_{t-1}. After the shift in x and y direction for all the corner points in the frame $f_{t-1\ is}$ found, the mode for all the shifts in x and y direction is found and that gives the net shift in x and y direction caused by panning of the camera.

The frame, f_t, is shifted by that much amount.

After shifting the frame f_t, a simple difference between the frame f_{t-1} and f_t gives the difference image which contains just the pixels (mostly on silhouette's) of the moving objects.

Algorithm for finding moving foreground objects

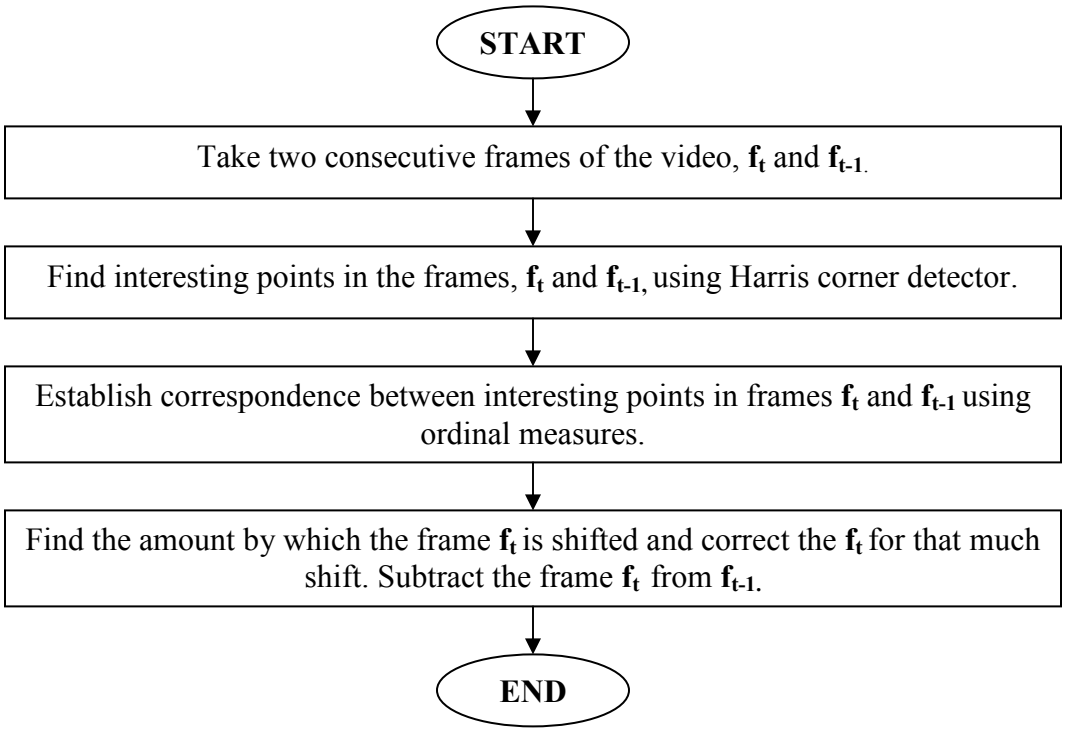


Figure 15. Block Diagram of Algorithm for finding pixels on moving foreground objects.

2.5 Results after Subtracting Background

The frames from 258 to 264 are shown below. During these frames the camera is panning towards right and the car is also moving towards right. The original frames are of the size 340*260 but have been scaled down for proper display. The frames from original 'car_pan.avi' video are shown in the 1st column and in the 2nd column the frames containing only the moving foreground objects (i.e. is only the car) are shown. Most of the pixels belong to the silhouettes of the moving object.

Frame258 	Frame258 
Frame259 	Frame259 
Frame260 	Frame260 

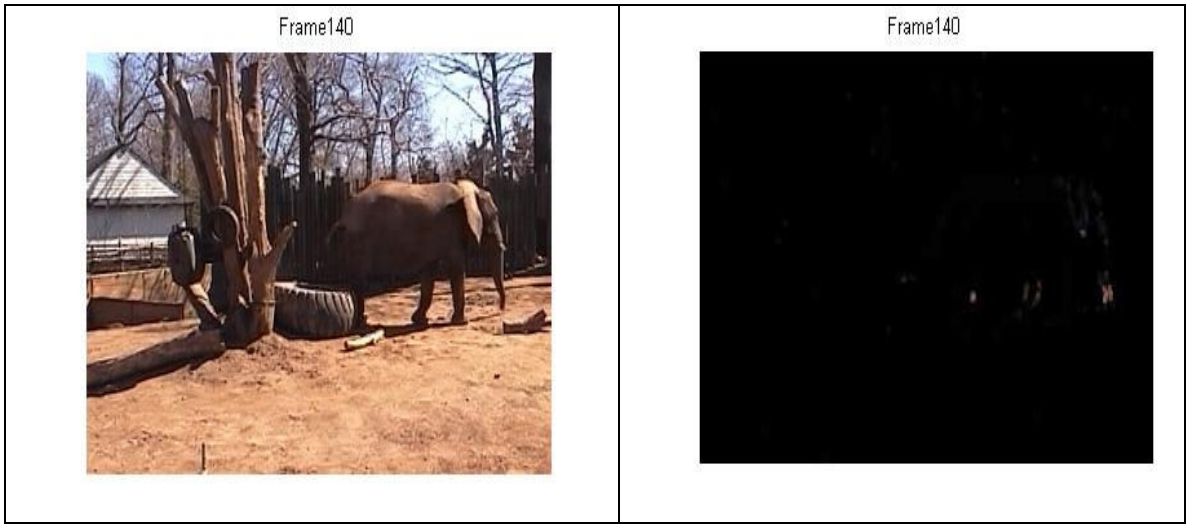
Frame261 	Frame261 
Frame262 	Frame262 
Frame263 	Frame263 

Figure 16. Results for Background Subtraction performed on the Video Taken from a Panning Camera.

The results for the videos taken from the stationary camera are also shown below.

The frames from 140 to 148 of the video 'Elephant.avi' (taken from [31]) are shown in 1st column. Only the elephant is the moving foreground object. Also the camera isn't panning.

The corresponding frames, containing the only the pixels on the silhouettes of the moving foreground objects and no background, are shown in the 2nd column.



Frame141 	Frame141 
Frame142 	Frame142 
Frame143 	Frame143 

Frame144 	Frame144 
Frame145 	Frame145 
Frame146 	Frame146 

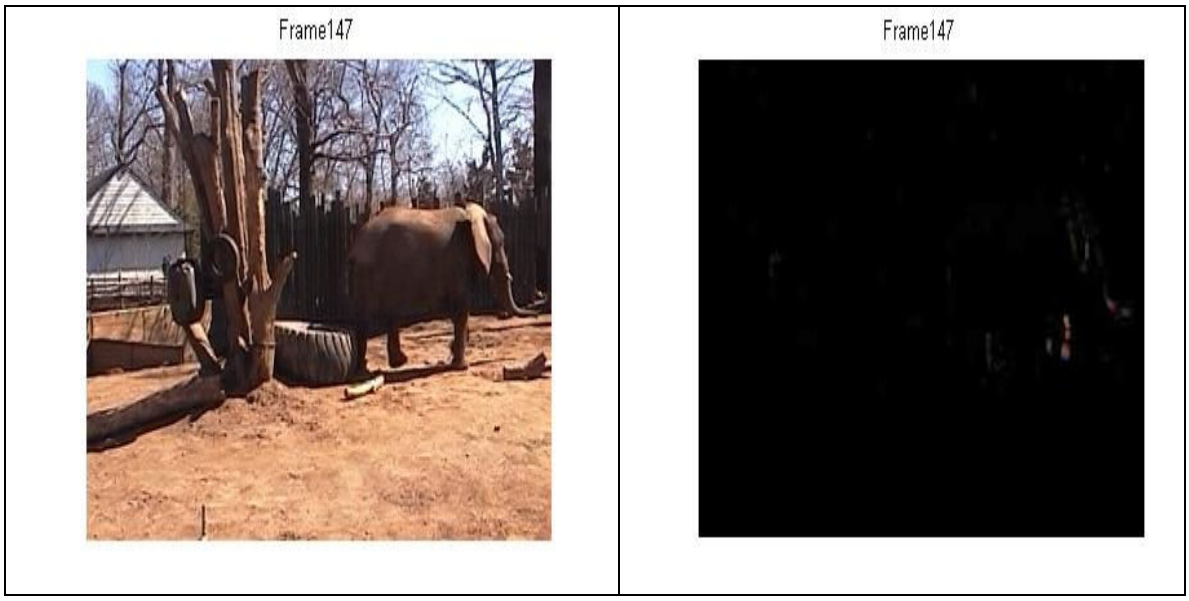


Figure 17. Results for Background Subtraction performed on a Video Taken from a Static Camera.

3 Object Segmentation and Related Work

3.1 Object Segmentation

From the previous section, after adjusting the current frame f_t, to correct for the motion introduce by panning of the camera, and subtracting it from the previous frame f_{t-1}, we get the pixels on the silhouette's of the moving foreground objects. The output of this stage is just the pixels on the silhouettes of moving objects. However the system doesn't know which group of pixels represents a single logical object. So the next step is to incorporate intelligence in the system so that it knows which groups of moving pixels are parts of a single logical object. For example, in a scene containing many moving cars there would be moving pixels on the front and back portion of every car and also a few pixels on the other boundaries of the cars. But the system should be able to tell which groups of moving pixels belong to which car.

Hence an ideal object segmentation system should be able to identify semantically meaningful components of the image and should be able to group the pixels belonging to such components.

A very detailed survey about different object segmentation techniques is given in [17]. Following sections summarize most of the relevant work given in [17].

3.2 Related Work

The main methods for segmenting moving objects are

1. Spatial Segmentation.
2. Motion based Segmentation.

Any monochrome image has two types of information: spatial information and intensity information. Spatial information gives the location of a pixel in the 2D image and intensity information gives the intensity or graylevel value of that pixel. The combination of the spatial and intensity information for regions of pixels defines the various features of an image like edges, corner, shapes etc. A video is a sequence of images collected at a certain interval of time. Examining a sequence of images yields a third type of information known as ***temporal information*** which can be defined as a combination of spatial and intensity information as a function of time. This work uses Spatio temporal information along with motion based information to segment moving objects.

3.2.1 Spatial Segmentation

The methods in this category directly manipulate the pixel intensities in the image plane or spatial domain of the current video frame to segment the objects. They don't use any temporal or motion information.

The methods in this category can be divided into 2 groups [17]

1. Region Based
2. Contour Based.

These methods are not very successful since they don't use temporal and motion based information present in sequence of frames of a video.

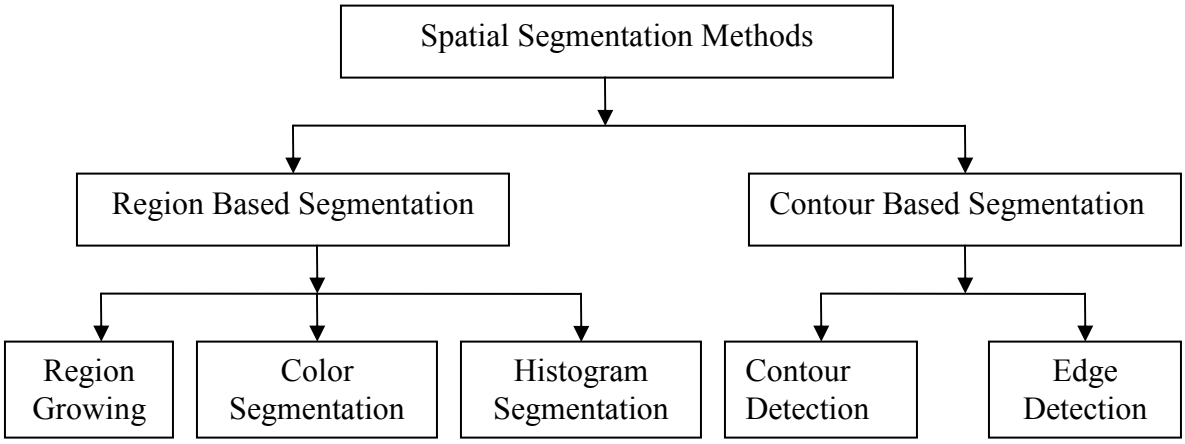


Figure 18 Classification of Spatial Segmentation Methods. Taken from [17]**.**

3.2.1.1 Region Based Method

The commonly used region based methods for object segmentation are

1. Color segmentation
2. Histogram segmentation and
3. Region Growing.

This section provides the overview of these three methods

1. **Color Segmentation:** Color segmentation is used to segment objects of specified color range from the input image. The range of colors to be segmented is specified and the algorithm iterates through each pixel in the input image to determine whether the pixel belongs to specified color ranges or not. ***Euclidean distance*** or ***Mahalanobis distance*** is used to find the distance between the color of the pixel and the desired color. More details about color segmentation are given in [9].

   But this is a very naive method for segmentation and not very useful. It is not possible to specify the ranges of the colors of all moving objects. Color composition for many moving objects cannot be predicted in advance. For

example a person might wear different color clothes on different days so it is difficult to predict the color of object in an image. Also many background and foreground objects may have the same color. In this case the background objects will also be classified as foreground objects.

So the color segmentation based methods are rarely used alone for segmenting moving objects. They are usually combined with motion based segmentation or edge detection.

2. **Histogram Segmentation:** Histogram is a graphical way of displaying the table which shows what proportion of cases fall into each of the several specified categories. In [28] histogram based segmentation is used. In [28] motion mask is obtained by using frame differencing, after correcting the frame for the motion of the camera. The motion mask obtained after subtracting the current frame from the previous frame generally contains the pixels on the silhouette of the moving objects. This motion mask is anded with the original frame to give the color of the deformable moving objects and along with it some portion of the background too. The histogram of the intensity value of the pixels obtained from anding mask and frame is then constructed. The threshold is applied to the histogram to get the most frequently occurring colors and the objects in the current frame with those particular ranges of colors are highlighted. But this method has few drawbacks. Coming up with a generic threshold for all the videos is difficult. If the threshold is not appropriate then this method may grab a considerable amount of background. Also if the moving object is of the same color as the stationary object then the stationary object will also be segmented. For example, if there are two polar bears of white color in the video and one is moving and the other is not, this method will segment both the moving as well as non-moving polar bears. This method doesn't take advantage of the motion information which can be obtained from the sequence of video frames.

3. **Region Growing:** Region growing is a technique to group pixels or regions into larger regions based upon a certain predefined criteria of growth. The main algorithm is to start with a set of seed points and from them grow the regions by adding to them the neighboring pixels having predefine properties similar to that of the seed [9]. For most of the segmentation application similarity criteria is the pixel intensity. It is known to have few drawbacks. The work [28] points out few drawbacks of this approach. In the work [28], the motion mask includes the pixel from foreground and some pixels from background too. Here there are chances that few selected seeds are actually part of background resulting in the complete segmentation of the whole background which is not desirable. Hence over-segmentation is always a problem in region growing based techniques.

3.2.1.2 Contour Based Segmentation

1. **Edge Detection:** Different edge detectors like Sobel or Canny edge detectors are used to find edges in the image. They are then used in combination with region growing to restrict the over-segmentation resulting from region growing.
2. **Contour Detection:** Closed contours are detected after detecting the edges in the image. The region within the closed contour is region grown with the region growing algorithm.

3.2.2 Motion Segmentation

The techniques in motion based segmentation falls into 2 groups depending upon the dimensions of the motion models used for segmentation. These groups are

1. 2D Approach and
2. 3D approach

The methods in 3D approach can be further classified into 2 categories

1. Structure from Motion
2. Parametric Methods.

A detailed explanation of motion segmentation is given in [17]. This section and its subsection summarize the work in [17].

3.2.2.1 2D Motion Based Segmentation

This category includes the methods which use ***Optical Flow*** and ***Change Detection*** for segmenting moving objects from the video. These methods are summarized in this section. More detailed information about these methods is given in [17].

1. **Methods based on Optical Flow Discontinuities:** Optical flow gives the apparent motion of the brightness pattern in the image. It is explained in detail in later sections. The method in this group uses displacement or optical flow of the image pixels to perform object segmentation. The optical flow of a pixel in an image is a velocity vector represented by the motion of the pixel in 2 consecutive frames. Different varieties of optical flows are given in [18], [19], [20] and [21]. The work in [17] gives detailed explanation about the different works using this method for object segmentation. In [22] optical flow is estimated based on features. They use corners computed by Harris Corner Detector as image features. If the actual distance between the corners in two consecutive frames is different than the amount by which camera shifts, then the corner is on a moving object, and its motion is the difference between its measured distance and the amount of shift produced by the camera. But feature based optical flow methods don't work well since the image features on the moving objects are sparse.

2. **Methods based on Change Detection:** Most of the methods using optical flow compute the velocity for each pixel in the frame. These methods are inefficient since most of the image point have zero velocity. Methods which compute optical flow for some image features like corners don't have good performance due to fewer features on moving objects. It is more efficient to compute the velocity of the pixels on the objects which are changing. Different background subtraction methods are used to get the foreground objects and only the pixels belonging to the foreground objects are used in computation of velocity.

3.2.2.2 3D Motion Based Segmentation

The 2D motion based methods only consider the motion of the objects on the 2D image plane and do not consider the real 3D motion of the objects [17].

The common methods for object segmentation which are based on 3D approaches are summarized in this section. Detailed information on these methods and the related work is given in [17].

1. **Structure from Motion (SFM):** Relative velocity of the moving objects computed from the 2D video frames is used to construct a 3D structure of the scene [17]. Methods based on ***Stereo Vision*** are used to obtain the 3D structure of the world from stereo images. Detailed explanation of these methods can be found in [23], [24] and [19]. The main assumption made in SFM is that the objects in a scene are moving rigidly or only the camera is allowed to move.

2. **Parametric Methods:** In SFM based methods, the algorithms work on a 3D scene having depth information such as stereo vision. But in parametric methods there is no assumption about the depth information. They explicitly assume the physical structure of the scene and use it to obtain the object motion in the 3D space. Unlike SFM they don't assume rigid body motion but they do assume piecewise rigidity [17]. A detailed description of these models and the related work done in that area is given in [17] and [19].

3.3 Approach taken in this work

In this work spatial, motion and temporal (i.e. information obtained from few sequences of video) information are used for segmenting moving objects. The block diagram of the whole approach is given in this section and individual stages are explained in detail in next sections.

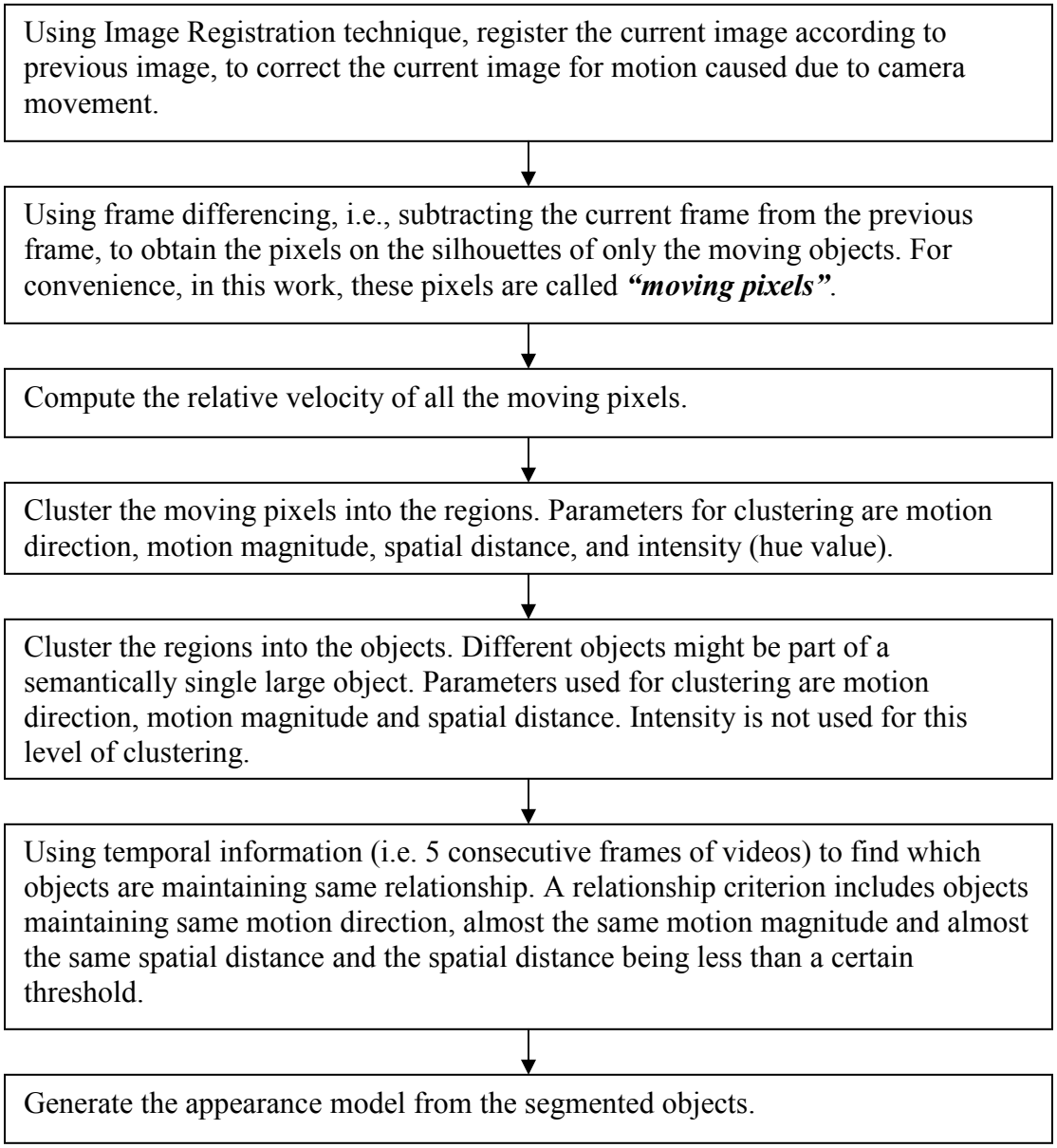


Figure 19. Block Diagram of the System.

Individual blocks are explained in detail in the later sections.

4 Finding Relative Velocity of the Moving Pixels.

4.1 Introduction

After adjusting the current frame f_t, to correct motion introduced by the panning camera, the current frame is subtracted from the previous frame f_{t-1} to get the moving foreground objects. While subtracting the frame f_t from the frame f_{t-1}, the pixels belonging to moving foreground objects have intensity greater than a certain threshold. These pixels are stored in a ***'motion'*** list. Mostly the pixels on the silhouettes of moving objects are obtained.

Hence the motion list is a one way link list containing the pixels on the moving objects in the current frame.

Further algorithm will work on this motion list rather than all the pixels of the current frame.

4.2 *'Motion List'* and its structure

The structure of motion list is as follows

```
typedef struct MList
{
        int row, col;
        int shiftRow, shiftCol;
        struct MList *next;
}motionList;
```

Member's of Motion List

1) col & row – x & y coordinates of the pixel respectively.
2) shiftCol & shiftRow – amount of motion in x & y directions respectively.
3) struct MList *next – pointer to the next node in the list.

One node is created for each moving pixel. ***'col'*** & ***'row'*** field of the node is set to the x & y coordinate of the pixel respectively. The fields ***'shiftRow'*** & ***'shiftCol'*** are not initialized because the motion of the pixel in x & y direction is not known when the motion list is created.

The 'shiftCol' and 'shiftRow' variables of the moving pixels are initialized to an impossible value of -1000.

The next step is to find the motion of the pixel in x & y direction.

4.3 Finding the relative velocity of the moving pixels.

The velocity of the moving pixels is used for segmenting the moving foreground objects.

A video is a sequence of frames captured in a fixed time interval. Hence each frame of the video can be considered as a function, ***I(x, y, t)***, of 3 variables of which ***x*** and ***y*** represent the spatial coordinate and t represents the time. ***Motion analysis involves recovering information about the motion in the scene from the variation of intensities I(x, y, t) over time*** [27].

The difference between Motion Field and Optical Flow is summarized in the table below taken from [26].

Motion Field	Optical Flow
1) It assigns a velocity vector to each point in the image.	1) It gives the apparent motion of the brightness pattern in the image.
2) It tells us how the position of an image of corresponding scene point changes over time.	2) Most of the time it is same as motion field. But in some situations they are not same.

But in some cases optical flow is not the same as motion field.

For example

1) **A Moving light**

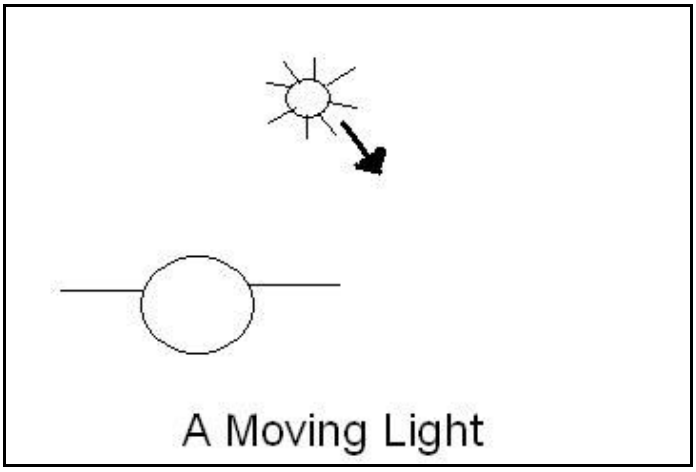


Figure 20. Object is stationary but the source of light is moving. Taken from [26].

The image changes so there is optical flow. But since the scene objects are not moving, there is no motion field.

2) **A Rotating Sphere**

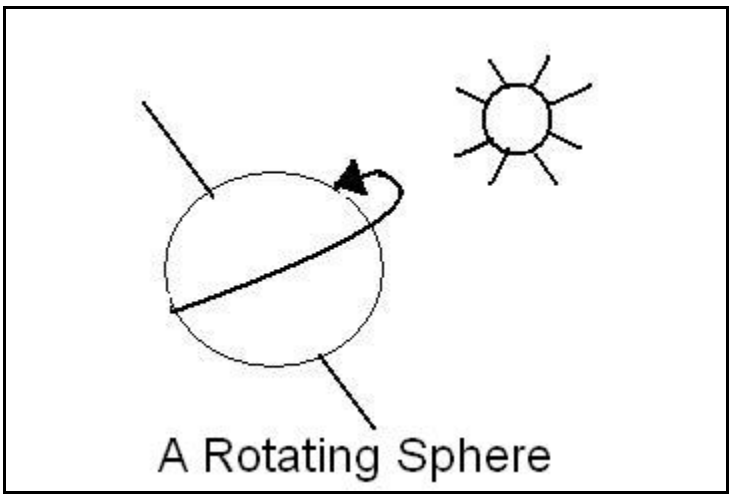


Figure 21. Object is moving but the source of light is stationary. Taken from **[**26].

The scene object moves so there is a motion field. But since the light is stationary there is no motion of the brightness pattern in the image and the image doesn't change. Hence there is no optical flow.

Barber's Pole Problem

It further explains that the optical flow and motion field are not always the same.

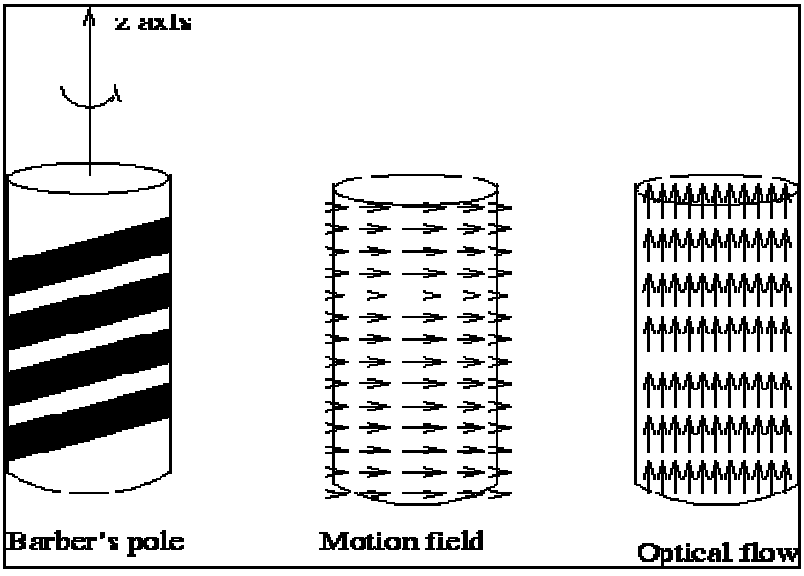


Figure 22. A cylinder rotating about its own axis. Taken from [25].

Since the pole is moving around its own z-axis, motion field is perpendicular to z-axis and since the shift in brightness pattern is along the z-axis the optical flow field is parallel to z-axis.

4.4 Approach used in this work for Finding Relative Velocity of the Moving Pixels.

The algorithm used in this work to find the velocity of moving pixels is

1) The current frame f_t is adjusted to compensate for the motion of camera.
2) When subtracting the frame f_t from f_{t-1} to get moving foreground objects, the motion list for the frame f_t is generated. Motion List nodes initially don't have any meaningful value for the members ***"shiftCol"*** and ***"shiftRow"*** because the amount of motion is not yet determined.
3) For each motion node in the frame f_t its 11*11 neighborhood is found and an intensity matrix of size 11*11 is generated giving the intensities of its 11*11 neighboring pixels. The rank matrix, '***m1'***, of size 11*11 is then constructed from that.
4) For that motion node, pixels in its 41*41 neighborhood in the frame f_{t-1}, are stored in a matrix of size 41*41, '***m2'***. This is based on the assumption that no pixel will have motion enough to move by more than 20 pixels in 2 consecutive frames.
5) For each pixel in ***'m2',*** pixels in its 11*11 neighborhood are ranked according to their intensities to get 11*11 rank matrix and then ***'ordinal measures'*** are used to compare that rank matrix with ***'m1',*** i.e., for each moving pixel in f_t a corresponding moving pixel in f_{t-1} is found using ordinal measures and then difference between their x and y coordinates are found which gives shifts in x and y direction for that moving pixel. In this way shifts in x and y direction for all the moving pixels in the frame f_t is found and their corresponding motionList node's ***'shiftCol'*** and ***'shiftRow'*** is set to the shifts in 'x' and 'y' direction respectively.

Algorithm for Finding Velocity of pixels in *'motion list'*

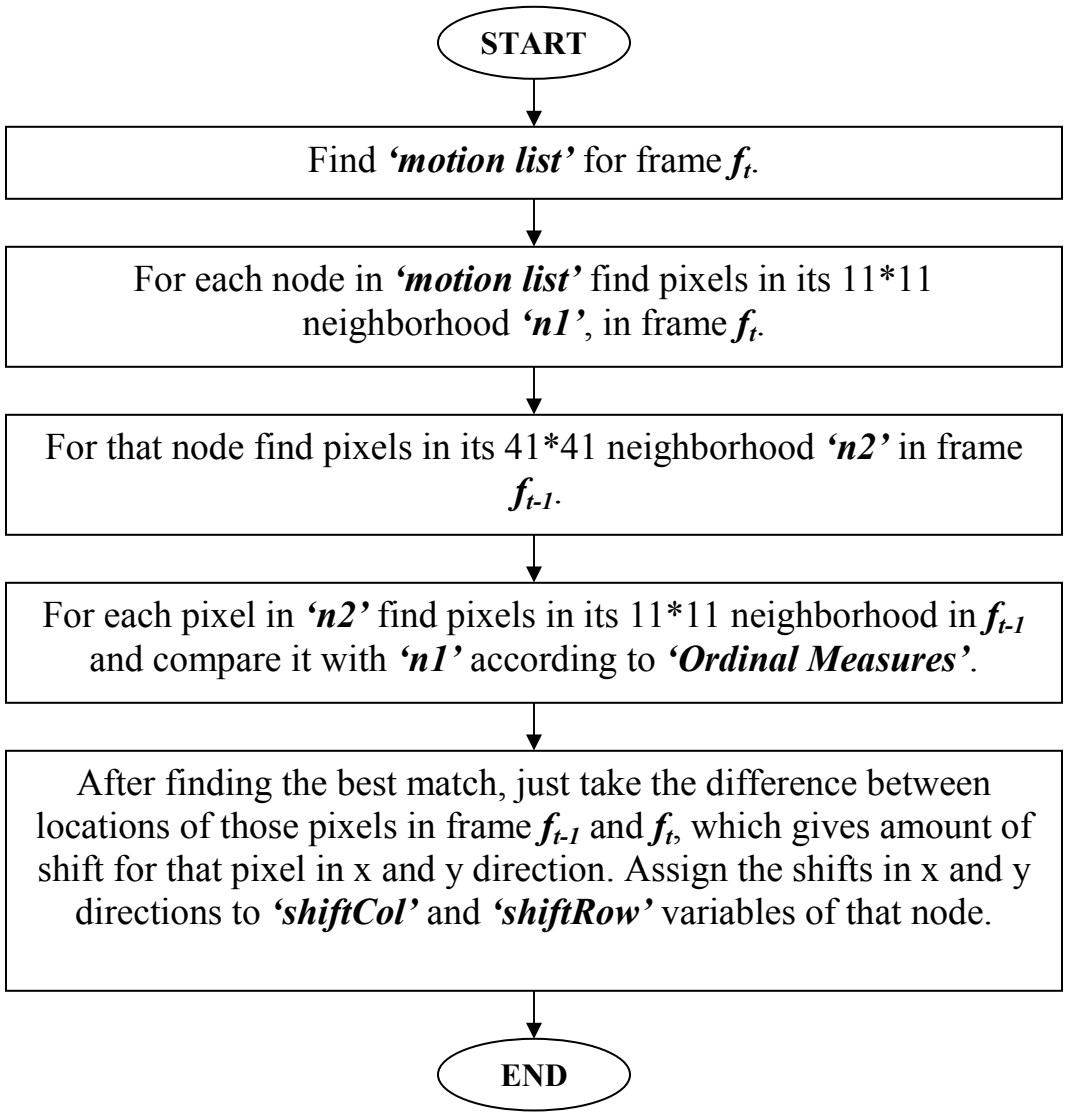


Figure 23. Block Diagram of Algorithm for Finding Relative Velocity of Moving Pixels.

4.5 Advantages of Generating Motion List

If the optical flow is implemented using Lucas-Kanade or any other methods then the above algorithm for pixel matching in two consecutive frames is done for all the pixels in the frames, which would be computationally very expensive. In any case before the velocity of moving pixel is found, the adjustment for moving camera has to be made. Hence when the frame f_t is subtracted from frame f_{t-1}, after correcting for camera

panning, the motion list is generated which has very few pixels or nodes as compared to the total number of pixels in the frame f_t and the above algorithm for finding velocity is applied only on this motion list. Hence the algorithm is comparatively fast.

4.6 Results

The frames from 116 to 124 taken from the video 'car_pan.avi' and having pixels with motion highlighted are shown below.

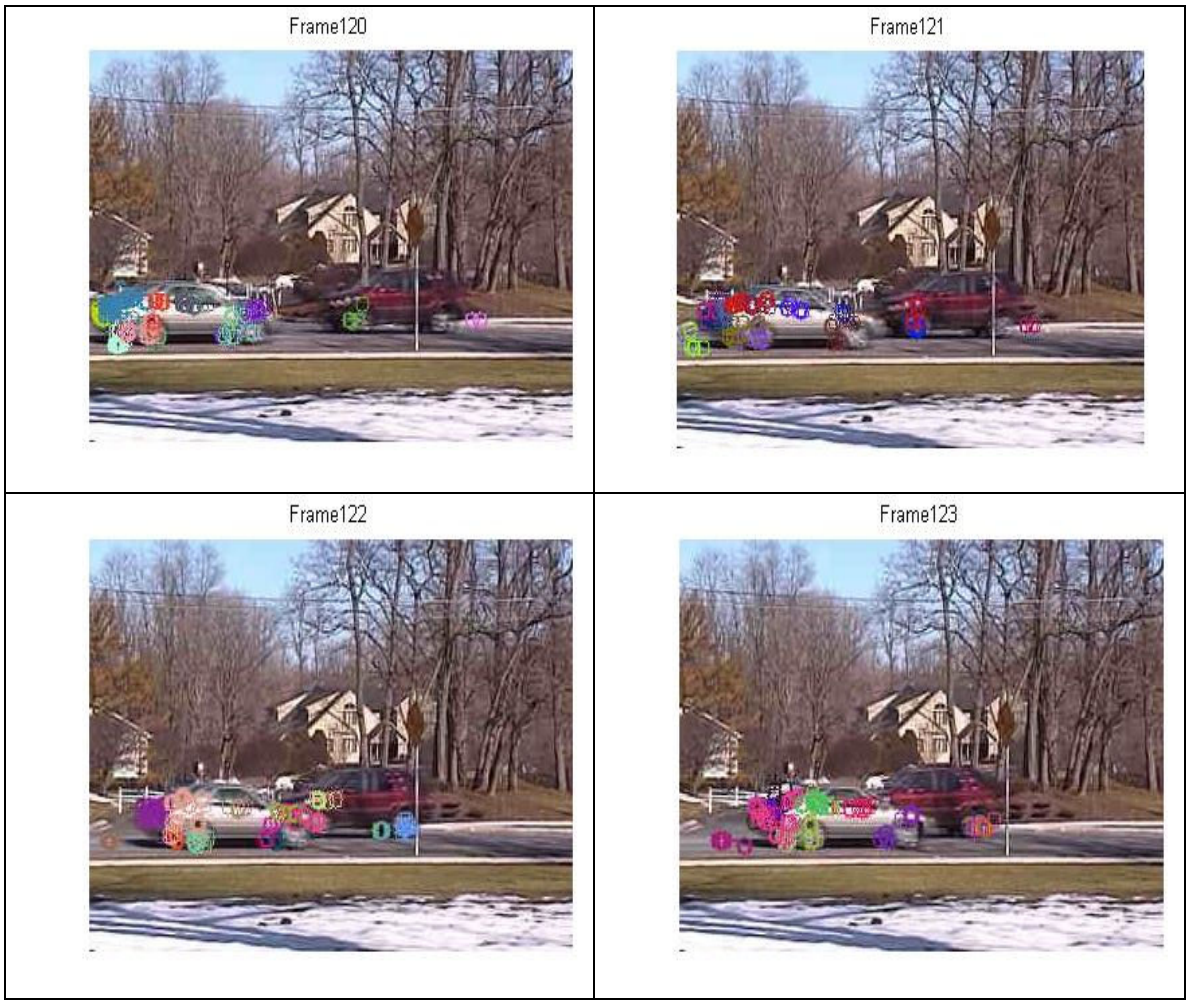


Figure 24. Results Showing Moving Pixels on the Video Frames.

5. Clustering Moving Pixels into Regions

5.1 '*Region List*' and its structure

After finding the moving pixels and their relative velocity and generating the ***'Motion List',*** the motion list nodes are clustered into the regions and a ***'Region List'*** is generated. The motion node contains just the moving pixels, most of them on the silhouettes of moving objects. There is no information about which group of motion pixel represents a single object. In order to segment the moving objects the algorithm should be able to determine which moving pixels are parts of a single moving object.

The structure of ***'Region List'*** is as follows

```
typedef struct RegionList
{
        int id;
        int objectMember;
        int avgIntensity;
        int xMotion, yMotion;
        int xMotionDir, yMotionDir;
        int row, col;
        int numPixel;
        int r, g, b;
        motionList *childStart, *childEnd;
        struct RegionList *next;
}regionList;
```

Members of region list are as follows

1) id – every region has a unique id
2) objectMember – the id of an object to which this region belongs to
3) avgIntensity – average intensity of the region
4) xMotion & yMotion – motion in x & y direction.
5) xMotionDir & yMotionDir - direction of motion in x & y axis. -1 in xMotionDir indicates all the motion nodes in the region are moving towards right and +1 indicates all the motion nodes in the region are moving towards left. -1 in yMotionDir indicates all the motion nodes in the region are moving down and +1 indicates all the motion nodes in the region are moving up.
6) col & row – x and y coordinates of the centroid of the region.
7) numPixel – total number of pixels in the region.
8) r, g & b – color coding for this region.
9) motionList *childStart, *childEnd – Many motion nodes are clustered into regions. So all the member motion nodes of a given region are stored in a link list whose head is motionList *childStart.
10) struct RegionList *next – pointer to the next region list node in the list.

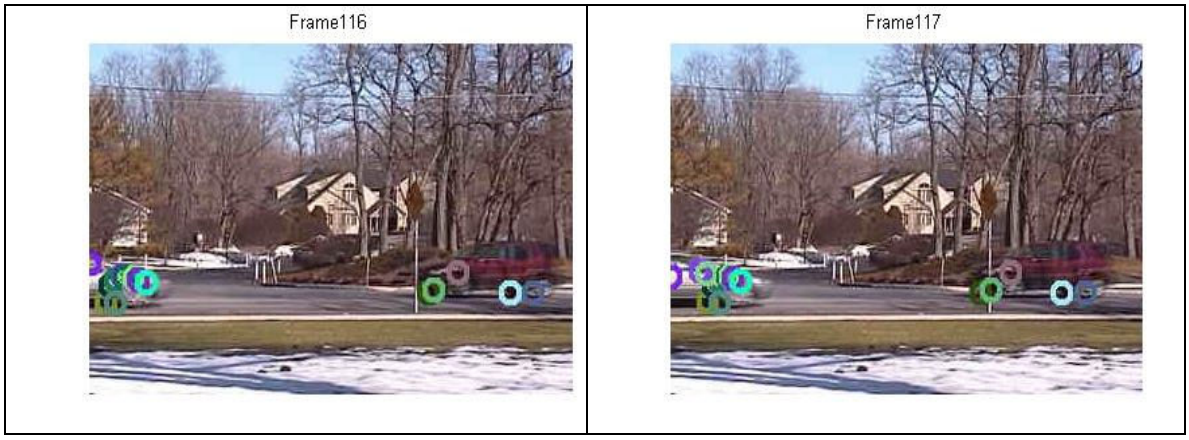
5.2 Parameters considered for generating region list.

The nodes in ***'motion list'*** are clustered into the regions to form a ***'region list'.*** Following criteria are used for clustering

1) Motion direction – This is the primary criteria in clustering. If 2 pixels have opposite motion direction they are not clustered in same region. All the pixels (or motion nodes) in the same region have the same direction of motion. It assumes that all the pixels in a rigid object will have same direction of motion.
2) Motion Magnitude – Any given region has pixels with almost same motion magnitude.
3) Distance – distance criteria is used to prevent grouping of the moving pixel with same motion direction, motion magnitude but very far from each other.
4) Intensity – Hue values (after converting image into HSV color space) of motion nodes in the same region is almost the same.

In a cluttered scene there may be easily hundreds of moving pixels (or motion nodes). The main idea of clustering the pixel (or motion node) into the regions is to group closely moving pixels with almost same intensities, almost same motion, and exactly same motion direction into one region. Then the further algorithm can work with ***'region list'*** which has relatively less number of nodes as compared to ***'motion list'***.

5.3 Results



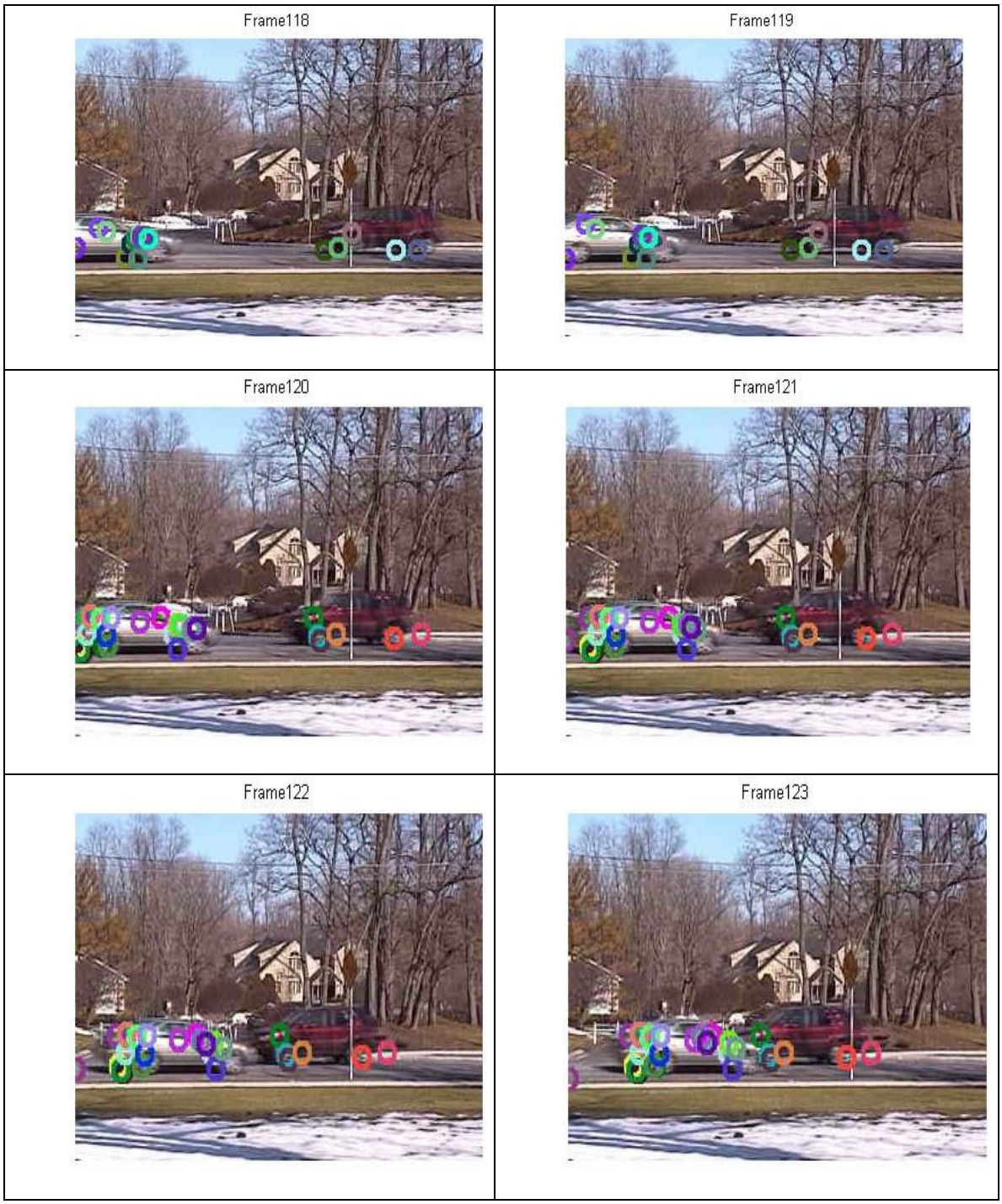


Figure 25. Results Showing the Regions on the Video Frames

6 Clustering Regions into Objects

6.1 Introduction

Regions formed by clustering the nodes in ***'motion list'*** and stored in a '***region list'*** are further clustered into objects, which in turn might represent a part of the complete object.

6.2 *'Object List'* and its Structure

The structure of ***'object list'*** is as follows

```
typedef struct ObjectList
{
        int id;
        int avgIntensity;
        int startFrame, endFrame;
        int width, height;
        int row, col;
        int numRegions;
        int totalXMotion, totalYMotion;
        regionList *regionStart;
        struct ObjectList *next;
        struct ObjectList *nextRelative;
}objectList;
```

Members of object list are as follows

1) id – every object has a unique id
2) avgIntensity – average intensity of that object.
3) startFrame – frame number when the object appeared in the scene
4) endFrame – frame number when the object left the scene
5) width & height – width and height of the object
6) col & row– coordinates of the centroid of the object
7) numRegions – total number of regions in the object
8) totalXMotion – sum of the motion of all the regions in X direction. When this is divided by the ***'numRegions'*** field it gives the average motion of the object in **X** direction.
9) totalYMotion – sum of the motion of all the regions in Y direction. When this is divided by the ***'numRegions'*** field it gives the average motion of the object in **Y** direction.
10) regionList *regionStart – Two or more regions are clustered into a single object according to clustering criteria. Hence every object has a list which contains its member regions. That list is pointed by ***'regionStart'***.
11) struct ObjectList *next – points to the next object in the list.

12) struct ObjectList *nextRelative – An object might be a part of a single large object. The objects are tracked for a certain number of frames and their relationship for that number of frames is observed. If a group of objects maintain the same relationship for a certain number of frames then they are considered to be the part of the same large object. Hence the ***'nextRelative'*** field of an object points to the list of objects, which have maintained same relationship with this object for certain number of frames. Hence an object along with the list of objects pointed to by its ***'nextRelative'*** field represents one single logical object.

6.3 Parameters Considered for Generating Object List

Regions are clustered according to the following criteria

1) Distance – regions which are at a distance less than a certain threshold are grouped into a single object; if they have the same motion direction in x and y direction and almost the same amount of motion.
2) Motion Direction – An object contains the regions having same motion direction along x and y axis. Regions with opposite motion direction are not clustered into the same object. This assumes that no two parts of the same object will have different motion direction.
3) Motion Magnitude – An object contains the regions with approximately same motion magnitude.

Intensity is not considered while clustering regions into objects because the same object might have different color regions.

6.4 Role of Intensity in Clustering Motion Nodes into Regions and Region nodes into Objects

Intensity (Hue value in HSV color space) is considered while clustering the motion nodes into the regions because the neighboring pixels with almost same intensities are grouped into one region. So the number of regions in the ***'region list'*** is very less as compared to the number of motion nodes in the ***'motion list'.*** So for further processing this region list is used, which, being smaller in size, is processed very fast.

Also there are chances that the algorithm for extracting the moving foreground objects might wrongly classify some pixels of the moving background as pixels of moving foreground objects and store it in ***'motion list'.*** So, if the background pixels have different intensity than the pixels of moving foreground objects, then those pixels would not be classified by taking into consideration the intensity factor. Also regions with number of motion nodes less than a certain threshold are discarded. This also helps in removing any outliers.

But while clustering the region nodes in the ***'region list'*** into objects, intensity is not taken into account, because a single moving object might have regions of different intensities.

6.5 Finding Relationship between Objects

After finding regions in the frame f_t and clustering them into objects, relationships between the objects are observed for certain number of frames.

Relationship criteria includes

1) Distance
2) Amount of motion &
3) Direction of motion

If two objects in the frame, f_t, maintain the same relationship for three consecutive frames f_{t-2}, f_{t-1} and f_t, then those two objects are declared as a part of same large object.

For this purpose objects found in a frame are stored in the ***'object list'*** and the last three objects lists are maintained. i.e., while processing the frame f_t, objects in it are found and stored in a link list whose head is given by ***'objectListHead[2]'*** and the objects of the last two frames, f_{t-1} and f_{t-2}, are stored in the ***'object list'*** pointed by ***'objectListHead[1]'*** and ***'objectListHead[0]'*** respectively.

The relationship is then established among the objects in frame f_t. i.e. if frame f_t has ***'n'*** objects then a 3 dimensional relation table of size ***n*n*4*** is created. The table has ***'n'*** rows and ***'n'*** columns because relationship is to be established among ***'n'*** objects and there is a depth of ***'4'***.

For example, for any row ***'i'*** and column ***'j'*** the entries of relation table are as follows

1) relationTable[i][j][0] = distance between object ***'i'*** & object ***'j'***.
2) relationTable[i][j][1] = difference between motion magnitude of object ***'i'*** & object ***'j'.***
3) relationTable[i][j][2] = difference between motion direction of object ***'i'*** and object ***'j'.*** It is 0 if both the objects have same motion direction and 2 if they have opposite motion direction.
4) relationTable[i][j][3] = relation count. If in the frames f_{t-2}, f_{t-1} and f_t the distance between object ***'i'*** and object ***'j'*** remains almost the same and is less than a certain threshold (to prevent objects moving far away from each other to be grouped together), the difference between the motion direction is exactly the same meaning that both the objects continue to move in the same direction and the difference between their motion is almost same, i.e., they have the same motion, then the relation count is incremented by 1. When the relation count reaches a certain threshold (in implementation three), the objects are declared as part of the same large object. Object ***'j'*** is added to the list pointed by the ***'nextRelative'*** field of object ***'i'***.

To summarize, if the object ***'i'*** and object ***'j'*** maintain almost the same distance which is within a certain threshold, exactly the same motion direction and almost the same motion magnitude for certain number of frames then they are declared as a part of same large object.

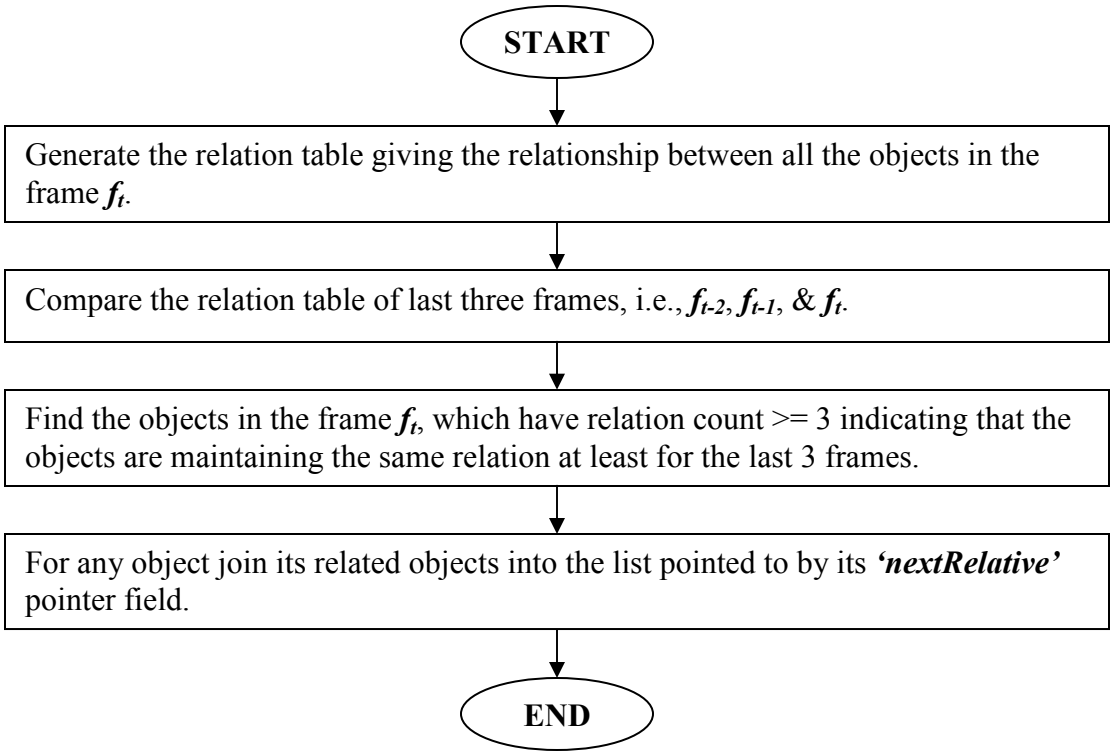


Figure 26. Block Diagram Showing Algorithm to Find Relationship among the Objects

6.6 Results

The results show the object segmentation in the frames from 123 to 130 of the video 'car_pan.avi'. These frames are typical since the two cars are going in opposite direction and are crossing each other and the algorithm successfully identifies them as two different moving objects. In any frame, different color rectangles represent different objects. The video is taken from a moving camera.

Figure 27. Result Showing Segmentation of Moving Cars Crossing Each Other

The results showing object segmentation in the frames 72 to 79 of the video 'PolarBear.avi', taken from [31], is shown below. These frames are typical because both

the polar bears are of same color, they are moving in the same direction, with almost the same amount of motion and also the distance between them is very less which is almost the same as their size. Initially for few frames, e.g., frame 72, the head of the rightmost bear and the body of the leftmost bear are grouped together as the same moving objects. But in the very next frame when the body of the rightmost bear comes in the view, the algorithm learns that there are two separate bears and henceforth successfully recognizes two bears as two different moving foreground objects as is shown by the rectangle of different colors.

Figure 28. Results Showing Segmentation of the Polar Bears

In the frames from 86 to 91 both the bears are completely in view. They both are successfully detected as 2 separate moving objects. These frames are typical because the bears maintain almost the same distance during the entire sequence, which is the same as their size and they are also moving in the same direction.

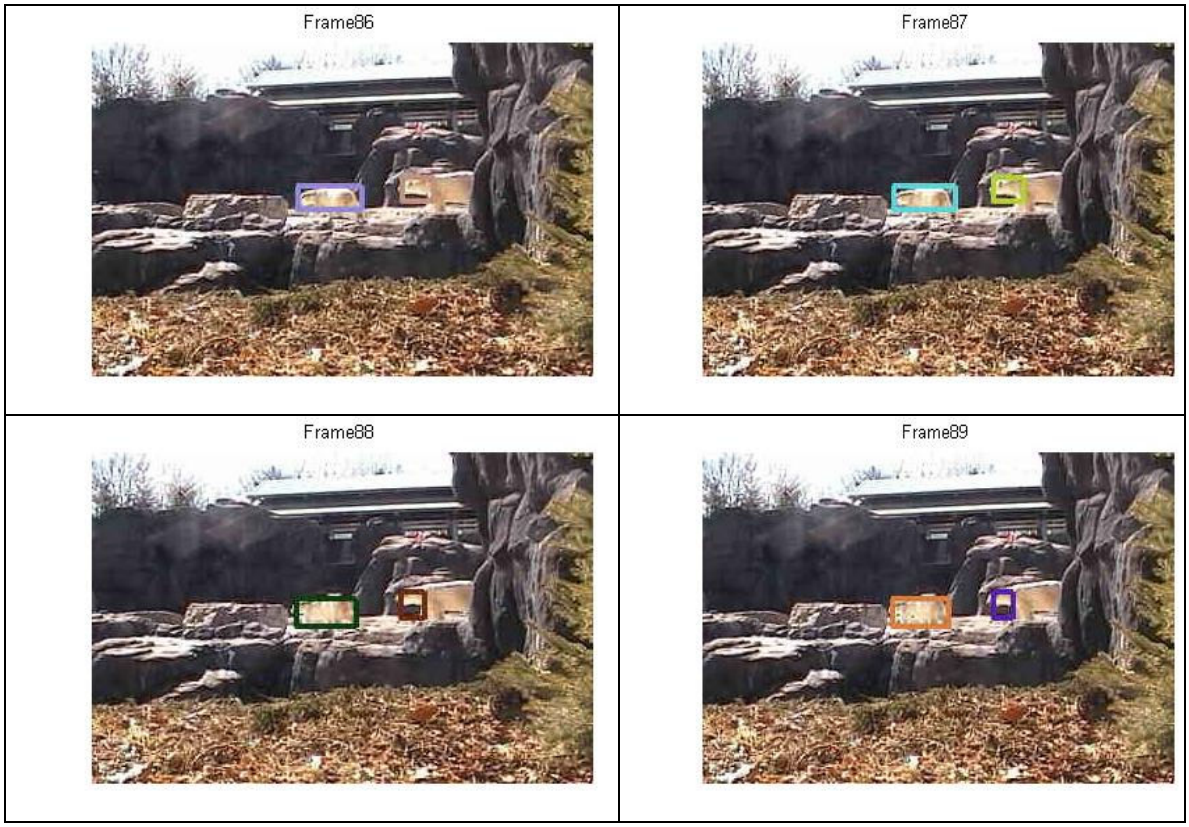
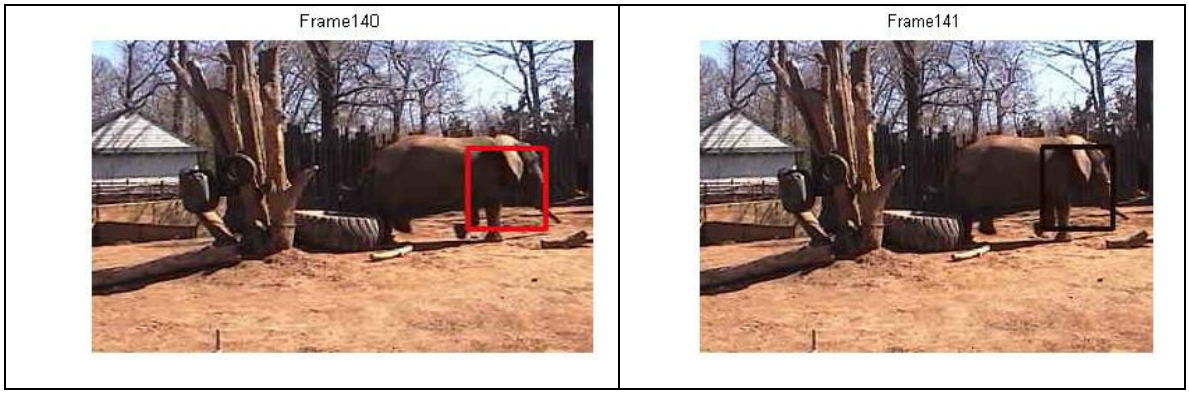


Figure 29. Results Showing Segmentation of Polar Bears.

The images below show the object detection done on the video 'Elephant.avi'. The frames from 140 to 147 of the segmented video are shown in fig [30]. These frames are typical because initially for few frames, up to 141, only the head and trunk portion of the elephant are moving and hence only they are grabbed as moving objects and not the whole elephant. After that, all of the elephant's body starts moving in synchronization and so the whole is grabbed as a single object from the frame 142 onwards.

The video is typical because of the similar texture and color of the ground and elephant.



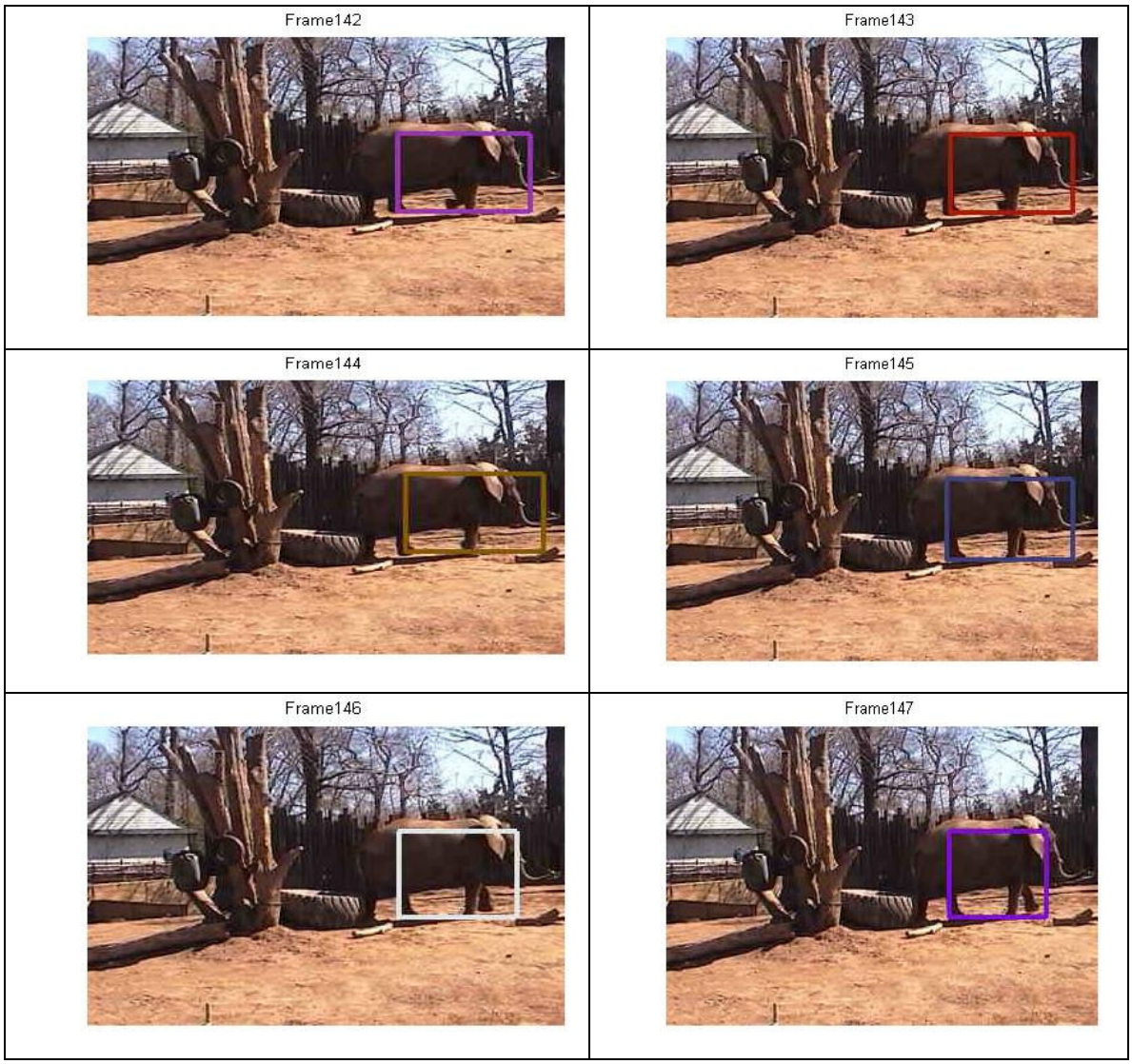


Figure 30. Results showing the segmentation of a moving elephant, having color and texture almost similar to that of the background.

6.7 Post Processing Methods to Improve the Results

The algorithm works very well for the moving rigid bodies. For example, for a moving car, the front and back portion of the car will have same amount of motion and same motion direction. So if the front and back portion of the same moving car are stored as different nodes in the ***'object list'***, then the algorithm will eventually learn that they represent the same object, since they maintain the same relationship for a certain number of consecutive frames.

But for non-rigid moving objects, different parts of the object might be moving independently. For example, a moving person might have his/her arms, legs, head, and

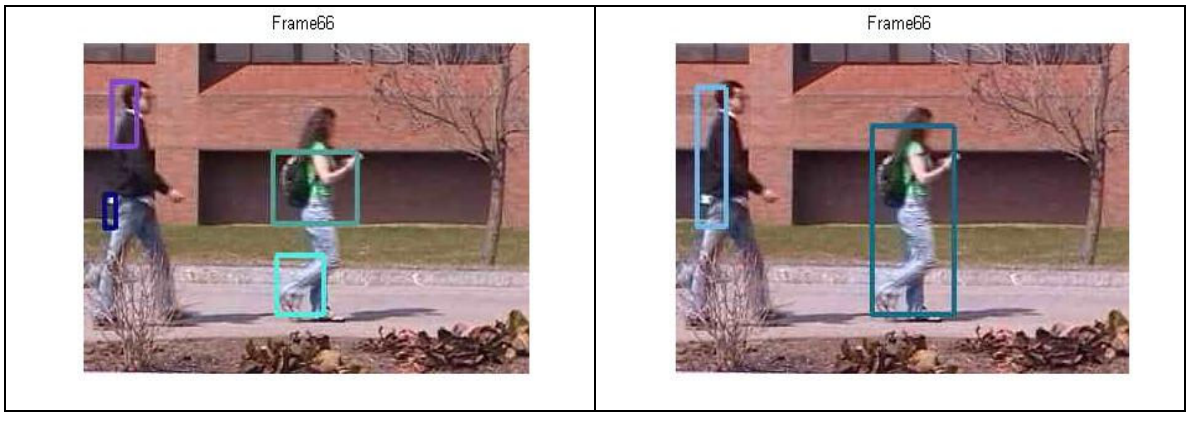
torso moving independently of each other, i.e., having same velocity but not maintaining a consistent distance. So the algorithm might end up segmenting arms, legs, head or torso as different objects. When an object is segmented a bounding box is drawn around it. So in the case of a moving person, if the arms, legs, head or torso are detected as different moving objects then the bounding box of these parts will be overlapping or might be very close to each other. Taking advantage of this situation, after determining the bounding box around the moving objects, the algorithm combines the overlapping or very close bounding box into single large bounding box. In the implementation two bounding boxes are merged together if:

1. The two boxes are overlapping and are having same motion direction.

2. If one bounding box is lying to the left or right of the other bounding box at a distance less than 30 pixels and is also lying either to the top or bottom of that bounding box at a distance less than 30 pixels and are having same motion direction.

It is very important to consider the motion direction while merging the two bounding boxes, to prevent the merging of two different objects moving in opposite direction and crossing each other. For example, two cars moving in opposite direction and crossing each other will have their corresponding bounding box overlapping each other for some frames, but they are not to be merged.

The results obtained after merging the boxes are shown in the fig [31]. The 1st column of the table indicates the resulting bounding boxes without merging and the 2nd column shows the resulting bounding boxes after merging. It can be clearly seen that there are different boxes around the legs, arms, head and torso in the 1st column and a single large bounding box in the corresponding row of the 2nd column.



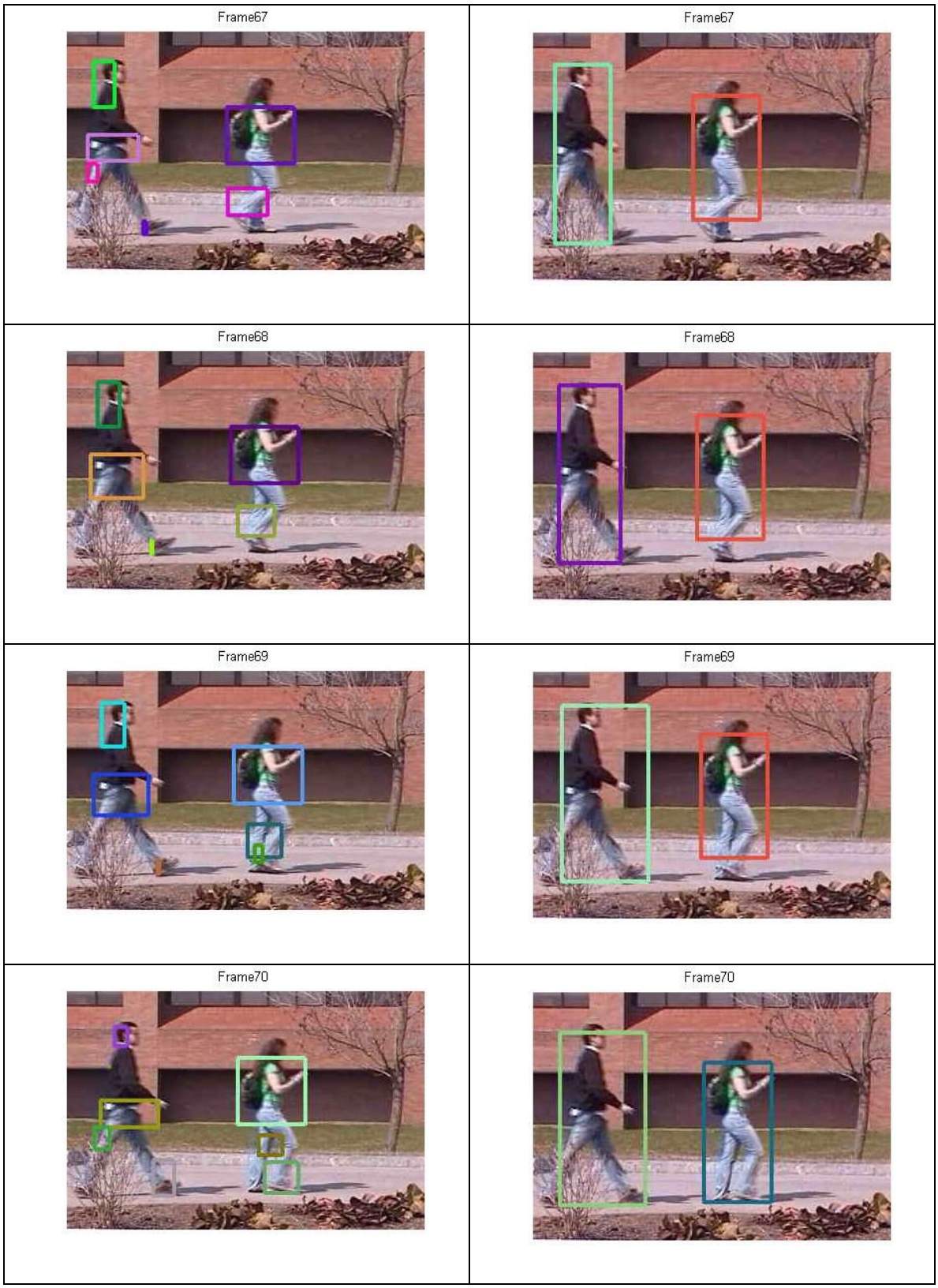


Figure 31. Results showing the effect of Box Merging method.

7. Generating Appearance Model of the Moving Objects

In section 4.2, the structure of objectList was explained in detail. One of the fields of the structure is the pointer ***"struct ObjectList *nextRelative"***, which is a pointer to the head of the link list of the objects which are related to the object. Every object might be a part of semantically single large object. The temporal information is used to find which objects maintain same relationship for certain number of frames and this information is used to group objects into a semantically single large moving object. Hence, if the group of objects maintains the same relationship for certain number of frames, then they are considered to be the part of the same object. Hence ***"nextRelative"*** field of an object points to the list of objects, which have maintained the same relationship with this object. Hence an object together with the list of objects pointed to by its ***"nextRelative"*** pointer represent a single large object.

Also an object has a pointer ***"regionList *regionStart"***, which points to the link list of the regions contained in that object.

Each region in turn has a pointer ***"motionList *childStart"***, which points to the link list of the moving pixels contained in that region.

So given an object, we can iterate through the list pointed to by its ***"nextRelative"*** pointer to find the objects related to it, and for all those objects we can iterate through their member regions using their ***"regionStart"*** pointer and then we can find the avgIntensity of the regions. The groups of average intensities of all the regions of all the related objects gives the ranges of the color present in that semantically single large moving object.

Each segmented object is enclosed in a bounding box. All the pixels in the bounding box are checked for the specified color. For example, if the specified color is black (R=25, G=25, and B=25), then all the pixels in all the bounding boxes are checked for having the specified color. Euclidean distance is used to compare the pixel intensity (in RGB color space) with the specified color and if it falls within a certain threshold (in this work 20), the pixel and the object containing it are highlighted.

7.1 Results for Generating Appearance Model

The 1st column of the table shown in the following figure, fig [32], shows the frames 64 to 67 from the object segmented video. The 2nd column shows that the objects having black color (R=25, G=25, and B=25) are highlighted and also the part of the objects having that color are highlighted.

The boy in the video is wearing a black color shirt and hence his shirt is highlighted. The girl in the video has a black color bag and hence only her black color bag is highlighted.

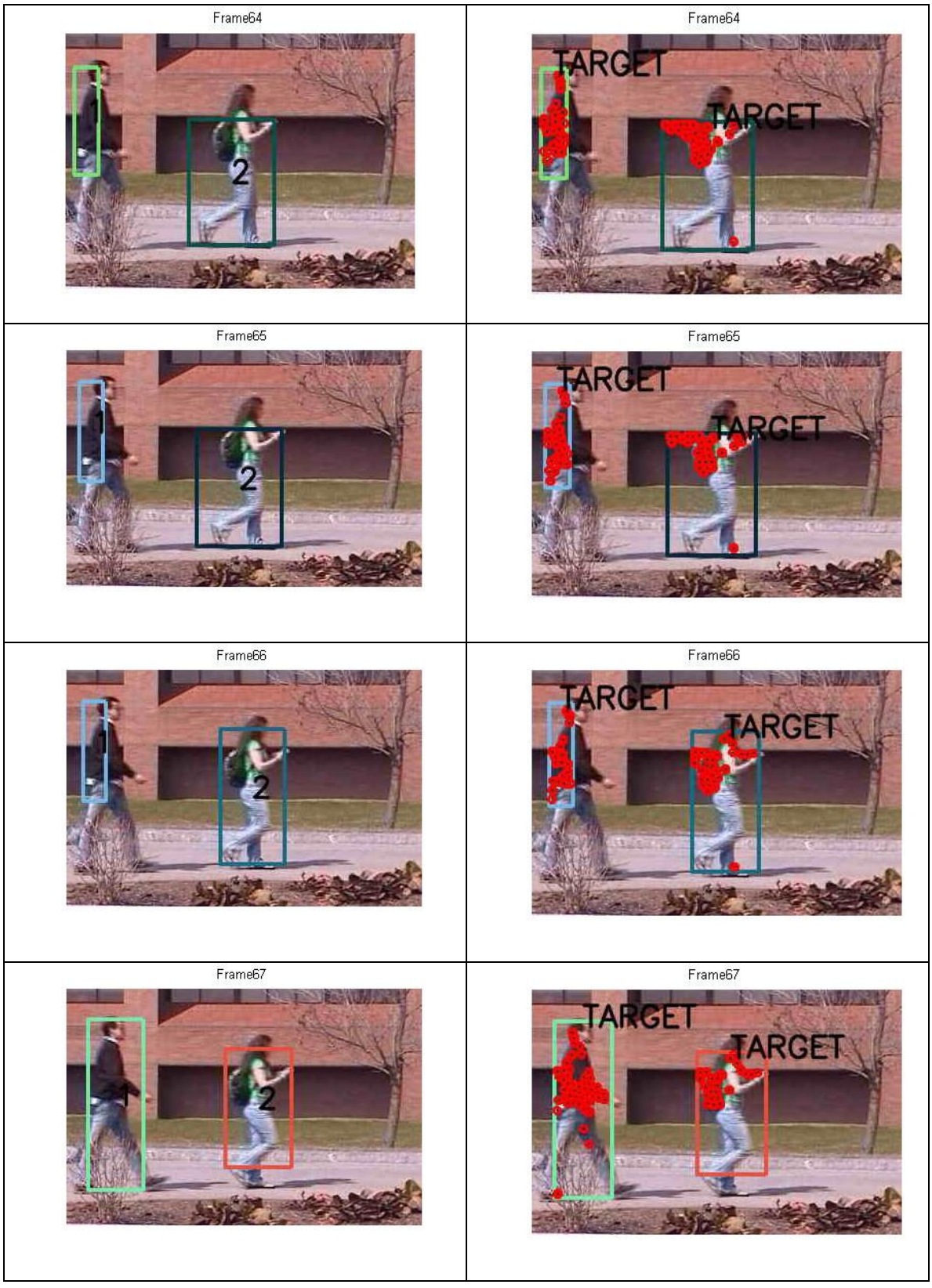


Figure 32. Results Showing the Selection of the Objects of Specified Color.

8. Extracting Spatial Relations between Any Two Objects in the Video

Extracting spatial relations among the moving objects in a video can be used for higher level understanding of the video. It can be used to annotate the frames of a video with meaningful labels such as “Two objects are meeting”, “One object is following another object”, “Two objects are moving away from each other”, etc. This semantic information can be used for searching the videos based on the queries such as “Find the videos in which two objects meet and then go away”, “Find the videos where one object overtakes or catches another object”, i.e., the queries based on the spatial relations among the objects. Object recognition system can be made on the top of the current system to answer queries like “Find the sequence of the videos in which a person walks in the middle of the room from west side”, “Find the video sequences in which Ferrari overtakes McLaren”, etc.

8.1 R-Histogram

R-Histogram, proposed by Yuhang Wang and Fillia Makedon [33], is used for finding the spatial relationships between the two objects. Angle histograms, proposed by Miyajima and Ralescu [34], can be used for finding the following directional relations between two objects:

1. Left of ,
2. Right of,
3. Above,
4. Below,
5. In front of, and
6. Behind.

R-Histogram extends the angle histogram by including the concept of ***labeled distance*** and is also able to find the following spatial relations in addition to the above directional relations:

1. Near,
2. Far,
3. Inside,
4. Outside,
5. Between,
6. Touching, and
7. Besides.

Suppose R a reference object and A is an object of interest. Then R-Histogram finds the spatial relation between R and A in the following way:

1. Consider a pixel x on the boundary of R and pixel y on the boundary of A. R-Histogram first uses the approach of Angle Histogram to find the directional

relationships between R and A. In constructing an Angle Histogram, angle of the vector from pixel x to pixel y, i.e. $\vec{xy}$, with respect to x-axis of the coordinate system is computed. Suppose the angle is denoted by $\theta(x, y)$. The range of the angle is [$-\pi$ to π]. The angle is computed for each of the pixel on the silhouette of R with every pixel on the silhouette of A. This set of angles gives the directional relation between R and A.

2. R-Histogram adds the concept of ***labeled distance*** to the Angle Histogram. The Angle Histogram doesn't take the advantage of the spatial distance between the reference object and the target object. The ***labeled distance,*** LD(x, y), in R-Histogram includes the spatial distance (Euclidean distance) between the pixel on the reference object and the pixel on the target object, d(x, y), and the label between the pixel x and y, l(x, y), which can be found from the table shown in fig [33].

LD(x,y) = (d(x, y), l(x, y))

Where LD(x, y) = Labeled distance,
d(x, y) = Euclidean distance between x and y and,
l(x, y) = label between x and y.

Pixel x inside A	Pixel y inside R	l(x, y)
False	False	0
False	True	1
True	False	2
True	True	3

Figure 33. Table showing computation of the labels in the labeled distance. Taken from [33].

3. R-Histogram also considers the set of vectors from each pixel on the boundary of R to every pixel on the boundary of A. Suppose x and y are the pixels on the boundary of R and A respectively. The bin H(I, J, L) of the R-Histogram is incremented according to the following equation [33]

$$H(I,J,L) = \begin{cases} H(I,J,L)+1 & if \theta(x,y) \in A_I \\ & \wedge\, d(x,y) \in D_J \\ & \wedge\, l(x,y) = L \\ \\ H(I,J,L) & otherwise \end{cases}$$

Where A_I is the range of angle values represented by the bin H (I, J, L), D_J is the range of distance value spanned by that bin and l(x, y) is the label in which the bin falls.

There are four quadrants, one for each label. For any vector $\vec{xy}$, the label is computed according to the table in fig [33] and then in the quadrant corresponding to that label, the bin corresponding to the angle between $\vec{xy}$ and x-axis of the coordinate system and the Euclidean distance between x and y is incremented by 1.

4. The histogram is normalized according to the following equation [33]

$$h(I,J,L) = \frac{H(I,J,L)}{\sum_{I'=1}^{n_A}\sum_{J'=1}^{n_D}\sum_{L'=0}^{3} H(I',J',L')}$$

8.2 Results for R-Histogram

The frame 33 of a video, 'walking.avi', is shown in the following figure, fig [34]. The frame shows two people moving towards each other.

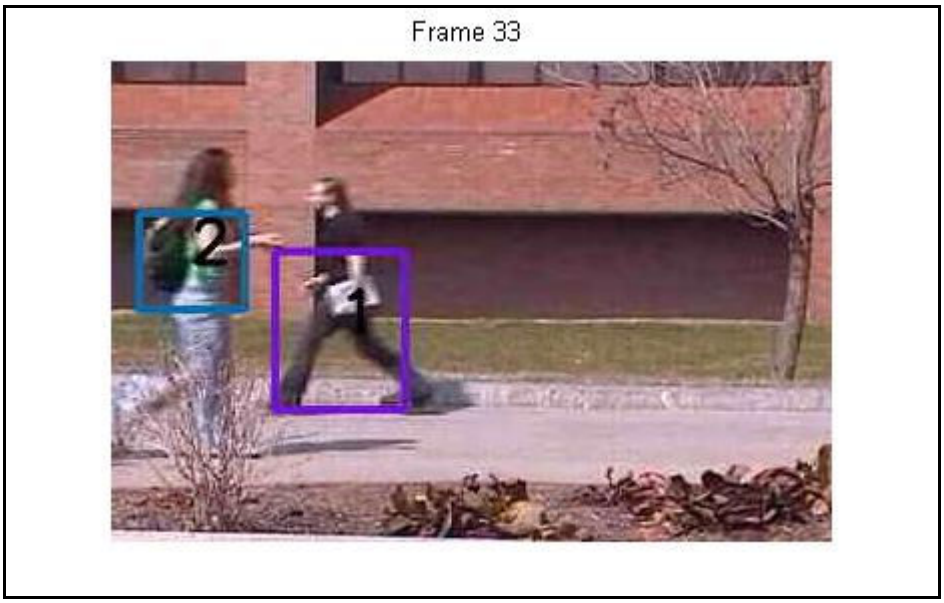


Figure 34. Frame 33 of a video showing two people walking towards each other.

Suppose the person labeled 2 is the reference object and the person with label 1 is the target object. Since both persons are at a certain distance from each other and none of their part overlap, only the bins in the quadrant corresponding to label 0 are incremented. The vector, from any pixel on the silhouette of person with label 2 to any pixel on the

silhouette of the person with label 1, makes an angle between $\pi/2$ to π with the x-axis of the coordinate system.

The fig [35] shows the four quadrants of the R-Histogram corresponding to four labels. The x-axis represents cos(angle), y-axis represents the Euclidean distance (divided into 10 bins) and z-axis represents the height of each bin after normalizing it.

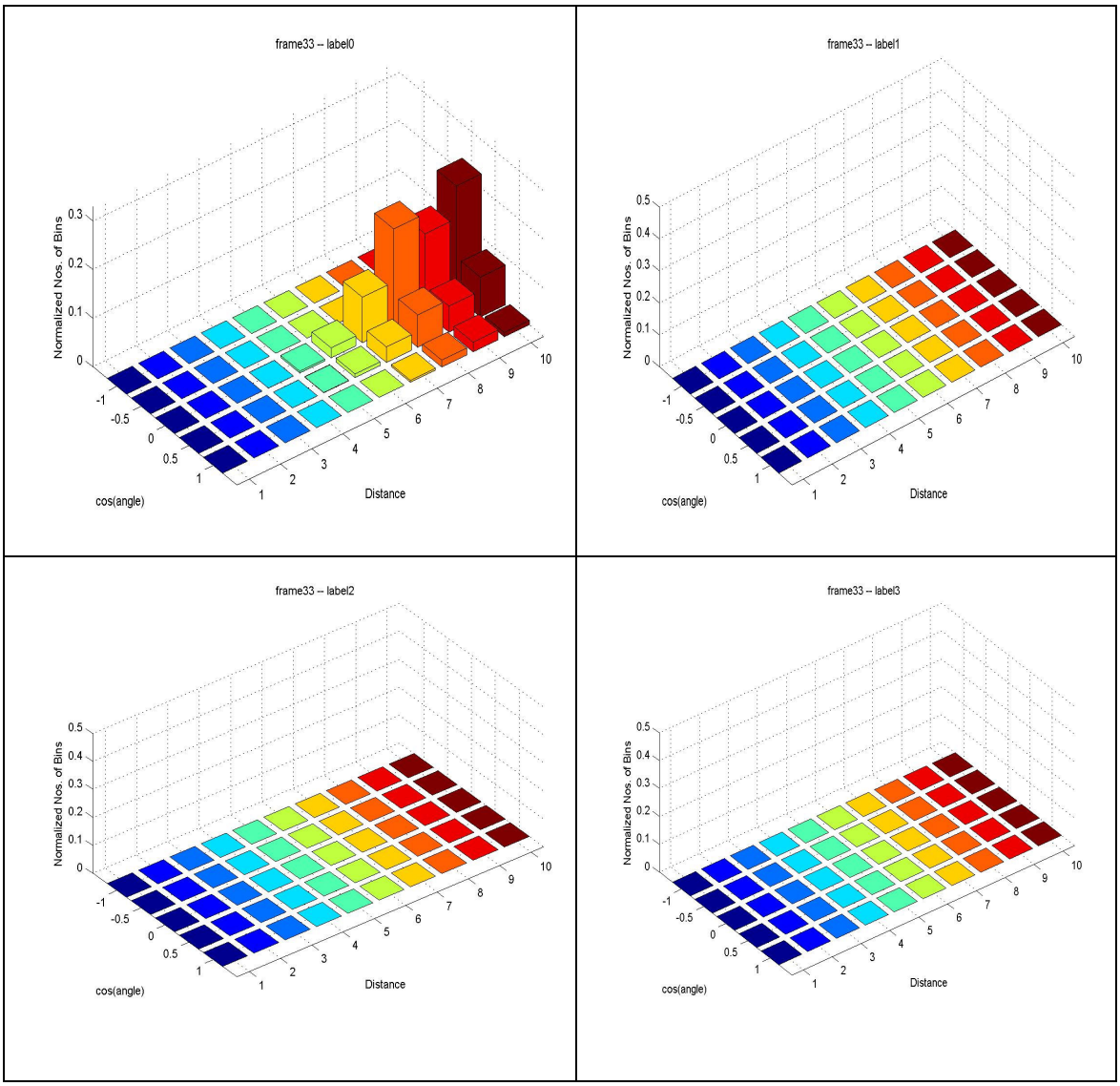


Figure 35. R-Histogram for the frame 33 shown in fig [34].

The frame 47 of a video, 'walking.avi', is shown in the following figure, fig [36]. The frame shows the two people crossing each other.

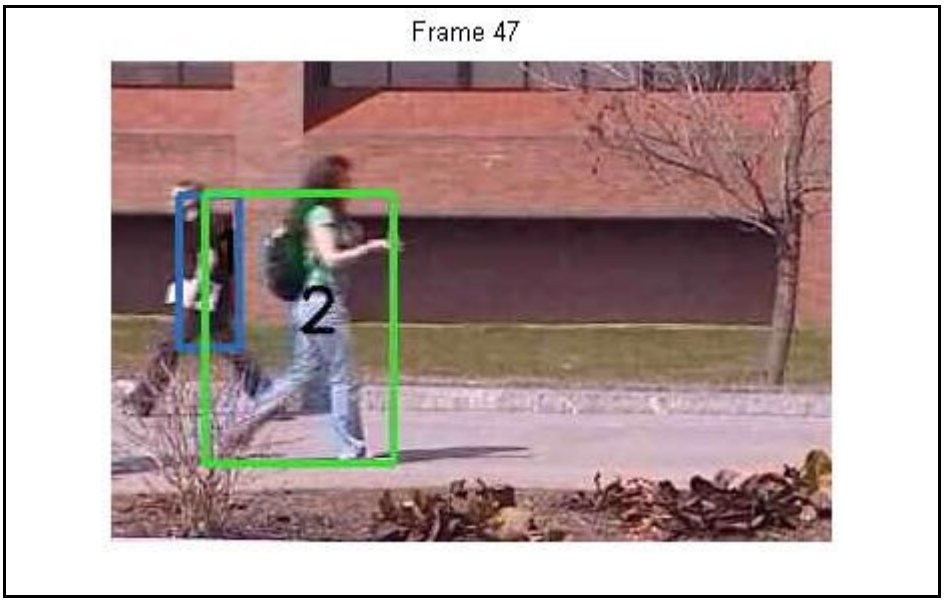


Figure 36. Frame 47 showing two people crossing each other.

Suppose the person labeled 1 is the reference object and the person with label 2 is the target object. The vector, from any pixel on the silhouette of person with label 1 to any pixel on the silhouette of the person with label 2, makes an angle between $\pi/2$ to π with the x-axis of the coordinate system.

There are many pixels on the silhouette of the reference object, e.g., the pixels on the silhouette of the boy's head, which don't lie inside the bounding box of object 2 and many pixels on the silhouette of the target object, e.g., the girl's hand, which don't lie inside the bounding box of object 1. Hence there are pixels in the quadrant corresponding to the label 0. There are many pixels on the silhouette's of the reference object, e.g., back leg and the back of the boy which lies in the bounding box of the object 2 and hence those pixels fall in the bin of the 3rd quadrant corresponding to the label 2. The R-Histogram is shown in the following figure, fig [37].

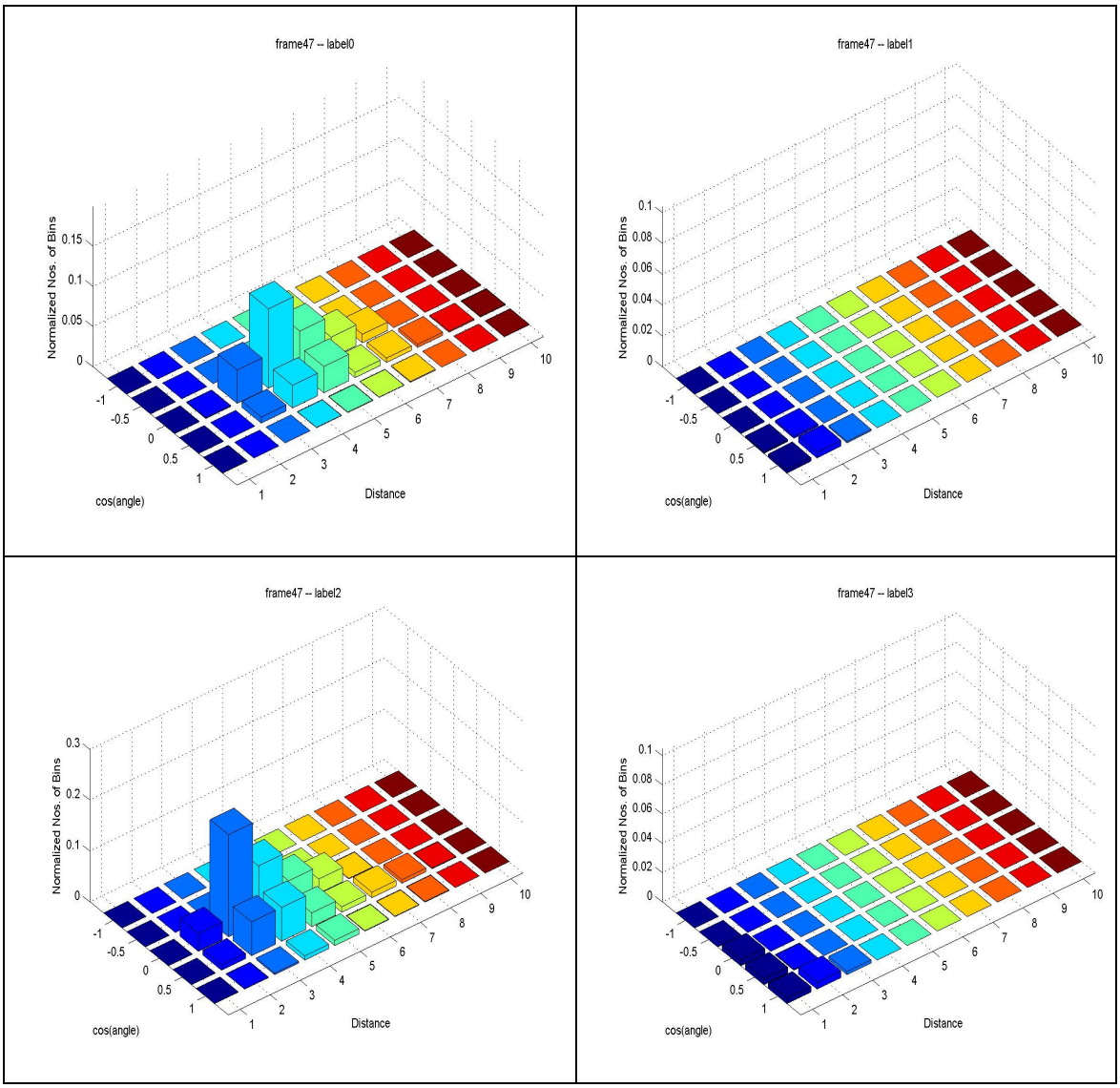


Figure 37. R-Histogram for the frame 47 (shown in the fig [36]).

8.3 Drawbacks of R-Histogram

The fig [38] shows the frame 70 of the video 'walking.avi'. The frame shows that the person with label 1 is following the person with label 2.

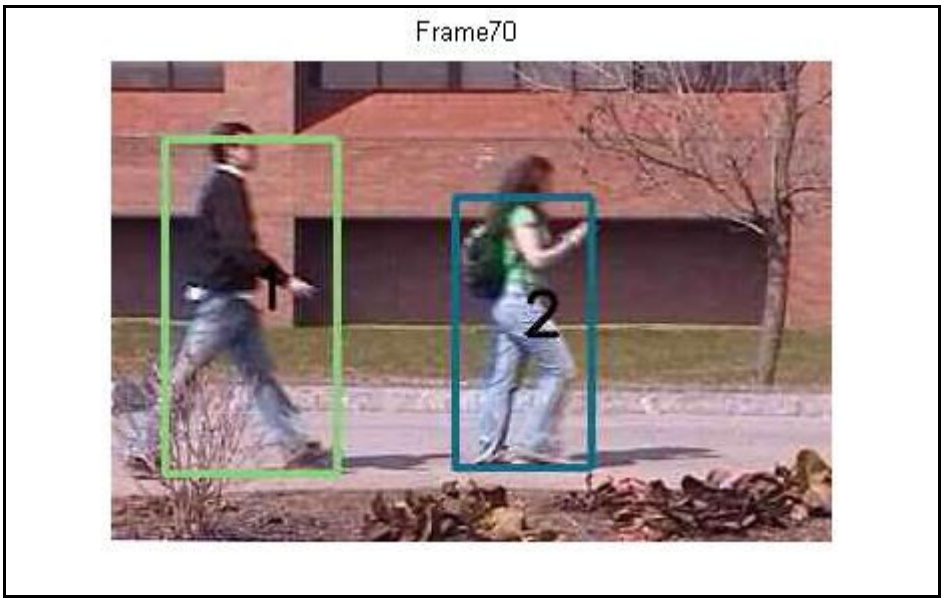


Figure 38. Frame 70 showing the person with label 1 following the person with label 2.

Suppose the person with label 1 is the reference object and the person with label 2 is the target object. The vector, from any pixel on the silhouette of person with label 1 to any pixel on the silhouette of the person with label 2, makes an angle between $\pi/2$ to π with the x-axis of the coordinate system

Since both persons are at a certain distance from each other and none of their part overlap, only the bins in the quadrant corresponding to label 0 are incremented.

The R-Histogram for this frame is shown in the following figure, fig [39].

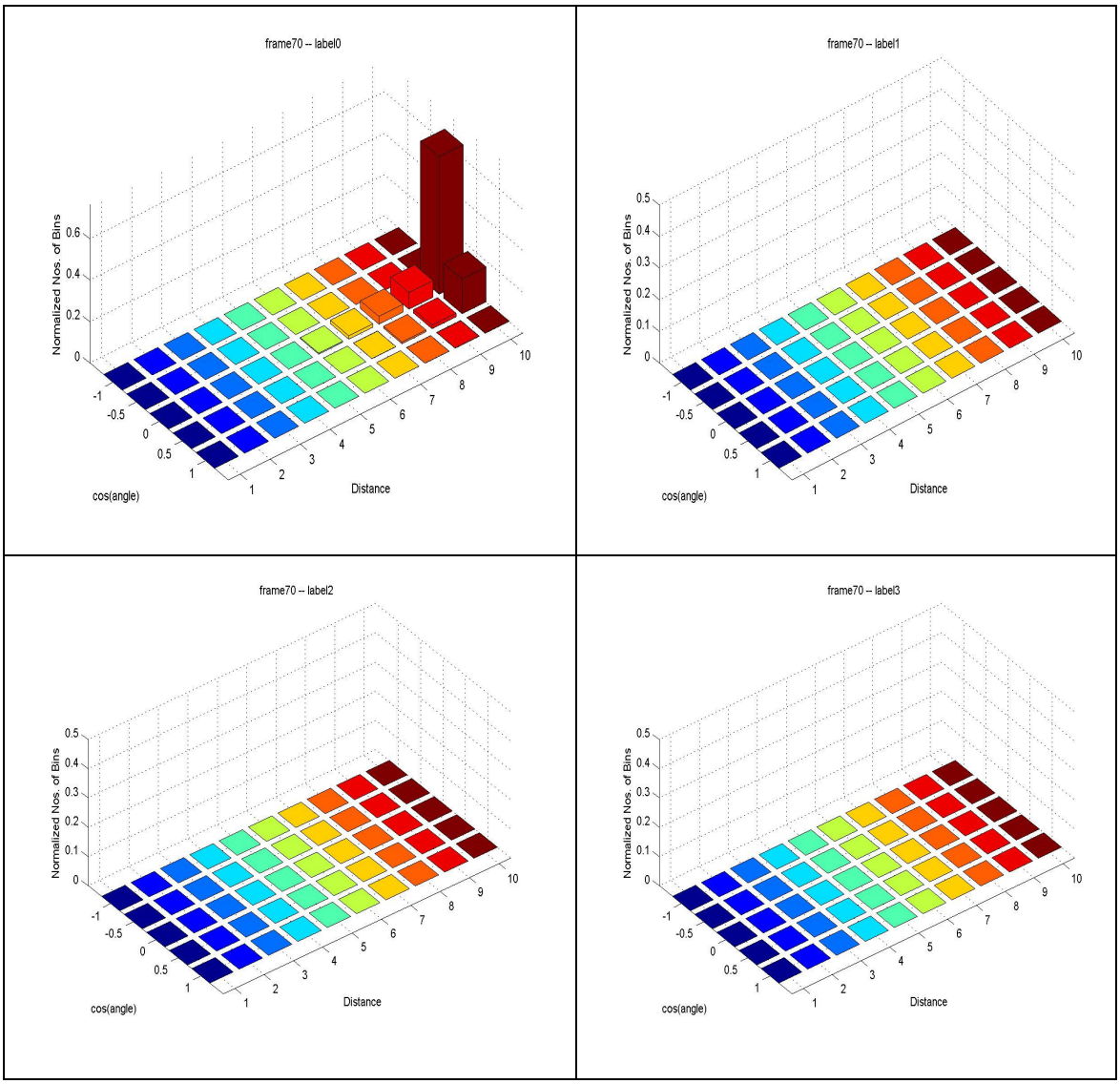


Figure 39. R-Histogram for the frame 70 (shown in fig [38]).

In the frame 33 shown in fig [34], the two persons were approaching each other and were certain distance apart from each other and still R-Histogram, as shown in fig [35], populated the quadrant corresponding to label 0. R-Histogram for the frames 33 and 70 are the same, even though, in frame 33 two people are moving towards each other and in frame 70 one person is following another person.

The R-Histogram is not able to distinguish among the cases such as two persons are moving towards each other or one person is following another person because it does not involves the direction information of the moving pixels.

8.4 Modified R-Histogram Used in this Work

In this work the x motion direction has been included in the labeled distance.

$$\mathbf{LD(x, y) = (d(x, y), l(x, y), m(x, y))}$$

Where LD(x, y) = Labeled distance,
d(x, y) = Euclidean distance between x and y and,
l(x, y) = label between x and y
m(x, y) = motion direction along x axis of the pixel x and y

The label between pixel x and y, l(x,y), is now calculated according to the table given in the following figure, fig[40].

Pixel x inside A	Pixel y inside R	XM(x) == XM(y)	l(x, y)
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Figure 40. Label in the modified labeled distance

Where XM(p) = motion in x direction of pixel p.

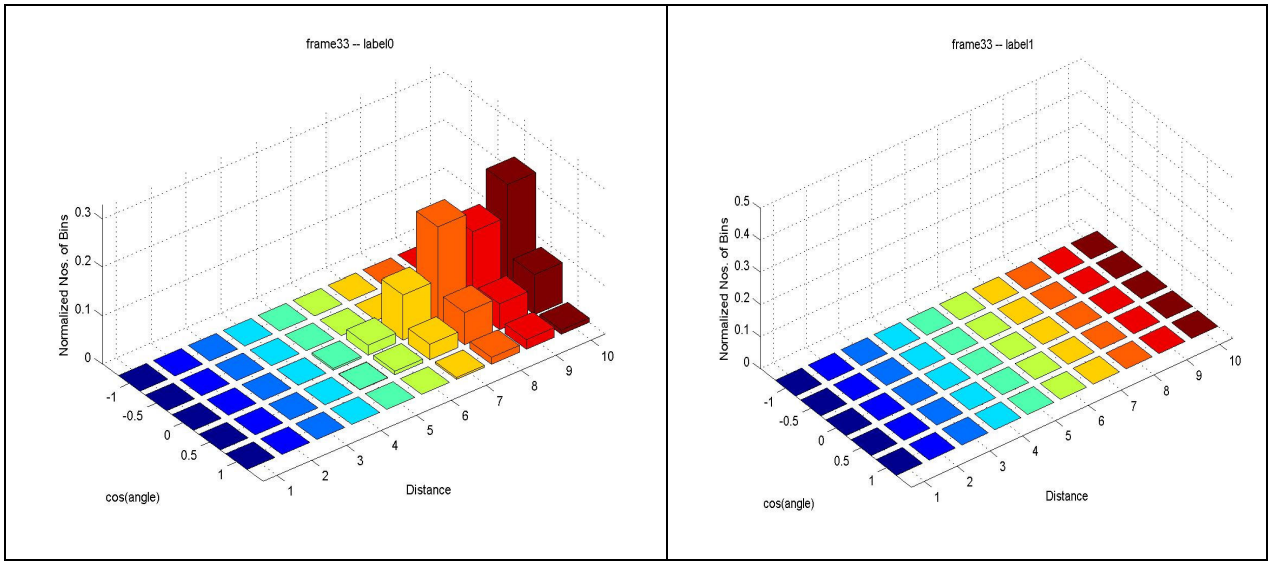
Now instead of four labels, as in the case of R-Histogram, there are eight labels. In addition to the spatial relations extracted by the R-Histogram, the modified R-Histogram proposed in this work is also able to identify the spatial relationships given in the following table in fig [41].

l(x, y)	Interpretation
0	The two objects are moving towards each other.
1	The Reference object is chasing or following the target object.
2	The two objects meet.
3	The reference object caught or overtook the target object.
4	The two objects meet.
5	The reference object caught or overtook the target object.
6	The reference and target objects are overlapping.
7	The reference and target objects are overlapping.

Figure 41. Interpretation of labels in the modified labeled distance

8.5 Results of Modified R-Histogram

The modified R-Histogram having seven labels was plotted for frame 33 (shown in fig[34]). As in frame 33 the reference and target objects are moving towards each other, i.e. their motion in x direction is not same and also they are certain distance apart. Hence the modified R-Histogram has pixels only in the quadrant corresponding to label 1 as shown in the following figure, fig [42].



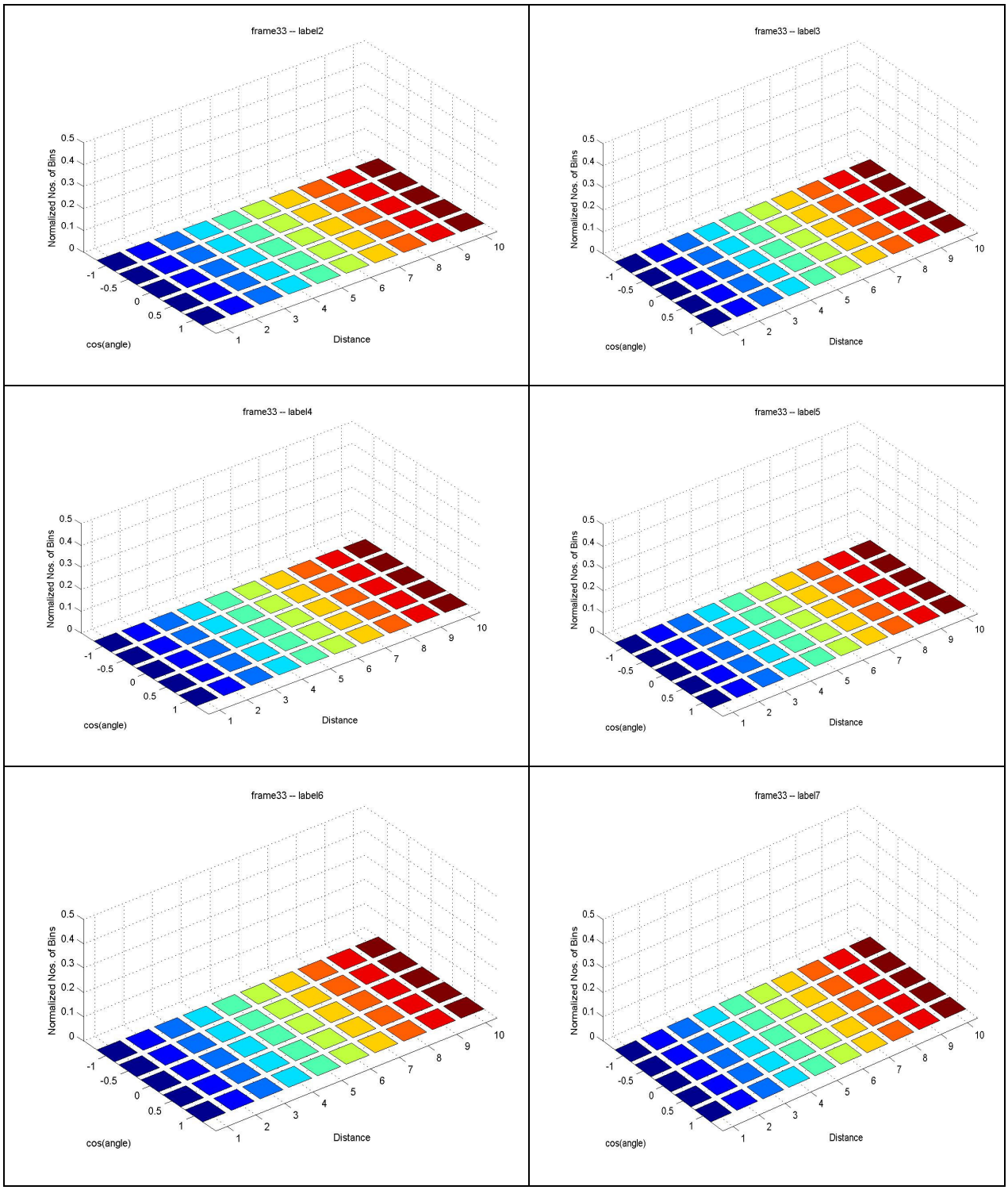
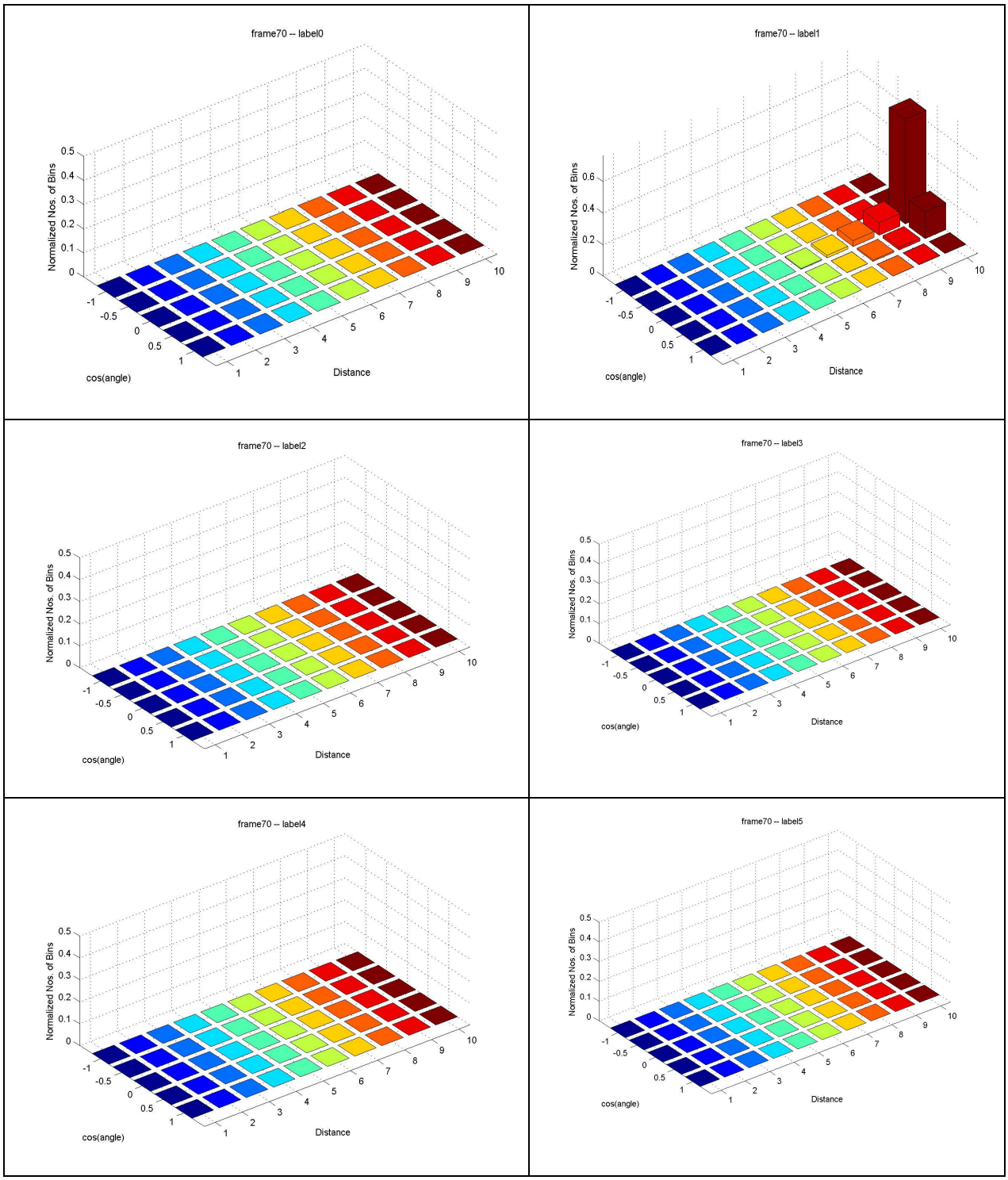


Figure 42. Modified R-Histogram for the frame 33 (shown in fig [34]).

For the frame 70 shown in the fig [38], the reference object is following the target object, i.e. their x motion direction is same. Also the reference object and target object are a

certain distance apart. Hence the modified R-Histogram places the pixels in the second quadrant corresponding to the label 1 as shown in the following fig [43].



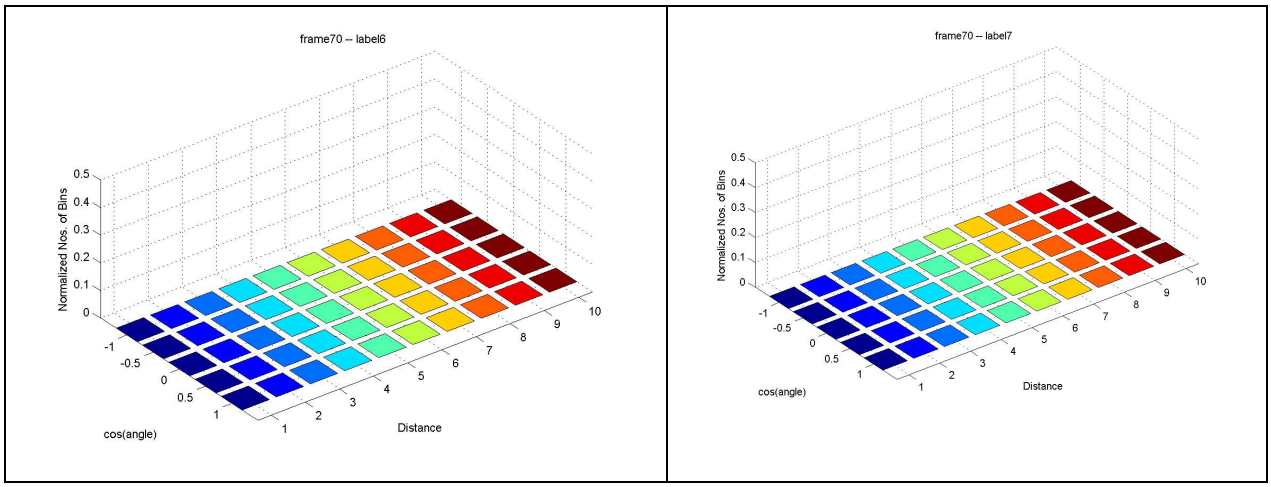


Figure 43. Modified R-Histogram for frame 70 (shown in fig [38]).

Hence the modified R-Histogram is successfully able to distinguish between the cases, when one object is following another object and the two objects are moving towards each other. It is also able to detect other spatial relationships in addition to the spatial relations detected by R-Histogram.

9. Conclusions and Future Work

The results shown in the previous sections prove that the approach suggested in this work successfully segments the moving objects and is able to generate spatial relations between any two moving objects and is successful in achieving higher level understanding of videos with respect to spatial relationships between any two objects. But there are some limitations to it. These limitations can be addressed in the future by extending this system.

1. If the two objects are moving in the same direction with the same amount of motion, side-by-side, then they would be detected as one single object, because the video is 2D and the depth information is lost when a 3D scene is projected on the 2D image plane. While finding the relationship between these objects, since they maintain same relative motion, same direction of motion and since the depth information is lost they also maintain the same spatial relation as they are moving side by side, they are segmented as same objects. Hence there is a parameter in the motion list called ***"depth"*** which is supposed to specify the depth of a moving pixel. In the future one can use stereo vision or other 3D motion and position estimation techniques to find the depth of the pixel. Hence the depth of the moving pixel can also be used as one of the clustering criteria which would prevent from clustering two objects moving side-by-side, in the same direction, with almost same motion.

2. The algorithm takes time proportional to the number of the moving objects in the scene. For a highly cluttered video, the algorithm takes a long time to find the moving objects and generate the appearance model from it. The algorithm for finding velocity of the moving pixels consumes a substantial amount of time, since for every moving pixel in the current frame it finds a matching pixel in the previous frame by searching its 41*41 neighborhood. In future new methods can be used to speed up the algorithm for finding velocity of the moving pixels.

3. In this work the spatial relation between two objects has been found. This system can be extended to find the spatial relationships among the group of objects.

4. An object recognition system can be made on the top of this system. After making object recognition system and extending the current work to find relation among the group of the objects, a system can be made which can answer queries like "find videos in which two person meet each other and go away", "find video sequences in which a person walks behind the cash counter", "find the video sequence in which one car overtakes another car" etc. The system can then answer the queries involving the spatial relations among the specific objects.

5. After object recognition, this system can also be extended to detect activities like fighting, hugging etc. So the videos can be searched for the queries like "find the video in which two or more persons are fighting".

6. Right now there is no object tracking in the system. The object tracking algorithm can be added to the system.

10. References

1. Stauffer, C. and Grimson, W. 1999. Adaptive background mixture models for real time tracking. In Computer Vision and Pattern Recognition.

2. Power, P. and Schoones, J. 2002. Understanding Background Mixture Models for Foreground Segmentation. Proceedings Image and Vision Computing New Zealand.

3. S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. "
Detection and location of people using adaptive fusion
Of color and edge information". In Proceedings of International
Conference on Pattern Recognition, 2000.

4. Brown, Lisa. "A Survey of Image Registration Techniques". ACM Computing Surveys. Vol. 24, No. 4, December 1992. Department of Computer Science, Columbia University.

5. http://en.wikipedia.org/wiki/Image_registration

6. http://mathworld.wolfram.com/AffineTransformation.html

7.http://www.cim.mcgill.ca/~dparks/CornerDetector/intro.htm

8. Bhat, Dinark; Nayar, Shree. "Ordinal Measures for Image Correspondence". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 4, April 1998. Columbia University.

9. Digital Image Processing Using Matlab – Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins

10. http://www.cee.hw.ac.uk/hipr/html/edgdetct.html

11. http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip-Derivati.html

12. http://www.netnam.vn/unescocourse/computervision/32.htm

13. http://oldweb.northampton.ac.uk/aps/eng/research/optophone/gaussian.htm

14. http://www.pages.drexel.edu/~weg22/edge.html

15. Carlo Tomasi . Convolution, Smoothing and Image Derivatives
http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/Image%20Processing.pdf

16. Videos for testing. http://www.ctr.columbia.edu/~dzhong/mtrack/demo.htm#Example

17. Segmentation of Moving Objects in Image Sequence: A Review – Dengsheng Zhang and Guojun Lu. http://personal.gscit.monash.edu.au/~dengs/resource/papers/cssp01.pdf

18. J.K. Aggarwal amd N. Nandhkumar. On the Computation of Motion from Sequences of Images - A Review. Proc. of IEEE, 76(8):917-935, 1988.

19. A. Murat Tekalp. Digital Video Processing. Prentice Hall PTR.

20. C.Stiller and J.Konard. Estimating Motion in Image Sequences: A tutorial on modelling and computation of 2D motion. IEEE Signal Processing. Magazine 16: 70-91, July 1999.

21. S.S. Beauchemin and J.L. Barron. The Computation of Optical Flow. ACM Computing Surveys, 27(3), Sept. 1995.

22. Smith, S.M. and Brady, J.M. "ASSET-2: Real-Time Motion Segmentation and Shape Tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, NO. 8, August 1995.

23. Daniel Scharstein. View Synthesis Using Stereo Vision. Ph.D thesis, Cornell University, 1997.

24. S.Geman and D.Geman. Stochastic relaxation, Gipps distributions and the Bayesian Restoration of images. IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-6(6): 721-741, 1984.

25. Owens, Robin. Optical Flow and Motion Estimation. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html

26. Image Processing and Interpretation at The University of Nottingham. Motion Estimation. http://www.cs.nott.ac.uk/~tpp/G5BVIS/pdf/2004/Lect7.pdf

27. Pinar Duygulu, Bilkent University. Motion Estimation. http://www.cs.bilkent.edu.tr/~duygulu/Courses/CS554/Spring2006/Notes/Motion.pdf

28. Sherif Azary. Detection of Deformable Objects in a Non-Stationary Scene. Masters thesis, Rochester Institute of Technology. 2005.

29. Parks, Donovan. Corner Detectors. http://www.cim.mcgill.ca/~dparks/CornerDetector/index.htm

30. OpenCV (**Open** source **C**omputer **V**ision library): http://www.intel.com/technology/computing/opencv/index.htm/

31. Dr. Roger S. Gaborski. www.cs.rit.edu/~rsg.

32. Fisher, Bob; Perkins Simon; Walker Ashley; Wolfart Erik. Gaussian Smoothing. http://www.cee.hw.ac.uk/hipr/html/gsmooth.html

33. Y. Wang and F. Makedon. R-Histogram: Quantitative representation of spatial relations for similarity-based image retrieval. In Proc. Of ACM Multimedia 2003, pages 323-326, Berkeley, California, USA, 2003.

34. K. Miyajima and A. Ralescu. Spatial Organization in 2D segmented images: Representation and recognition of primitive spatial relations. *Fuzzy Sets and Systems, 65(2-3): 225-236, 1994.*